

Operating System/2 Programming Tools and Information Version 1.2/1.3

Presentation Manager Programming Reference Volume 1 and Volume 2 Technical Upgrade

85F1529

Operating System/2
Programming Tools
and Information
Version 1.2/1.3

Presentation Manager Programming Reference
Volume 1 and Volume 2
Technical Upgrade

First Edition (December 1990)

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

COPYRIGHT LICENSE: This publication contains printed sample application programs in source language, which illustrate OS/2 programming techniques. You may copy and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the OS/2 application programming interface.

Each copy of any portion of these sample programs or any derivative work, which is distributed to others, must include a copyright notice as follows: "© (your company name) (year) All Rights Reserved."

© Copyright International Business Machines Corporation 1990. All rights reserved.
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Special Notices

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

IBM
IBM C/2
IBM COBOL/2
IBM FORTRAN/2
IBM Macro Assembler/2
Operating System/2
OS/2
Presentation Manager
Systems Application Architecture.

The following terms, denoted by a double asterisk (**) in this publication, are trademarks of other companies as follows:

Adobe	Adobe Systems Incorporated
Helvetica	Linotype
Microsoft	Microsoft Corporation
PostScript	Adobe Systems Incorporated
Times New Roman	Monotype

© Copyright IBM Corp. 1990

About This Book

This book, as well as the other books in the upgrade package, contains programming information that is new to or different from the information in the *IBM* Operating System/2* (OS/2*) Programming Tools and Information Version 1.2* library (part number 6024929). In addition, information from the *OS/2 Programming Tools and Information Version 1.2 Technical Update* (part number 64F1705) is included in this upgrade package.

The upgrade describes features added by the OS/2 Version 1.3 product, and amends some of the information that was published with OS/2 Version 1.2.

How To Use This Book

This book changes the information contained in the following books:

Book Title	*Book Number*
Presentation Manager Programming Reference Volume 1	64F0276
Presentation Manager Programming Reference Volume 2	64F0277

Only those chapters of the Version 1.2 books that are changed by the Version 1.3 product are included herein. The chapter numbers and titles are the same as those in the Version 1.2 book. As a convenience, the beginning of each chapter in this book provides a summary of changes for that chapter. Where this book changes information that is contained in the Version 1.2 books, page references are given for those books.

Who Should Read This Book

This book is for application designers and programmers who are using the components of the *IBM OS/2 Version 1.2/1.3 Programming Tools and Information Technical Upgrade* package. The reader is assumed to be familiar with the services of OS/2.

* Trademark of the IBM Corporation

© Copyright IBM Corp. 1990

Contents

Volume 1: (Data Types and Function Calls) 1

Chapter 2. Data Types 3
BITMAPINFO 3
BITMAPINFOHEADER 4
DRAGIMAGE 4
DRAGINFO 5
DRAGITEM 5
DRAGTRANSFER 7
ESCSETMODE 8
FONTMETRICS 9
HFILE 15
HSTR 15
MARGSTRUCT 16
MLE_SEARCHDATA 16
MT 16
OVERFLOW 17
PAPSZ 17
PRDINFO 17
PRDINFO3 18
PRIDINFO 19
PRJINFO 19
PRJINFO2 20
PRJINFO3 22
PRQINFO 24
PRQINFO3 26
RENDERFILE 27
SPLERR 27

Chapter 3. Device Function Calls 29
DevEscape – Escape 29
DevOpenDC – Open Device Context 29

Chapter 3.1. Drag Function Calls 31
DrgAcceptDroppedFiles – Direct Manipulation for Files 32
DrgAccessDraginfo – Access Drag Information 34
DrgAddStrHandle – Create String Handle 35
DrgAllocDraginfo – Allocate DRAGINFO Structure 36
DrgAllocDragtransfer – Allocate DRAGTRANSFER Structures 37
DrgDeleteDraginfoStrHandles – Delete DRAGINFO String Handles 38
DrgDeleteStrHandle – Delete String Handle 39
DrgDrag – Drag 40
DrgDragFiles – Begin Dragging Files 42
DrgFreeDraginfo – Free DRAGINFO structure 44
DrgFreeDragtransfer – Free DRAGTRANSFER Storage 45
DrgGetPS – Get Drag Presentation Space 46
DrgPostTransferMsg – Post Drag Message 47
DrgPushDraginfo – Access a DRAGINFO Structure 48
DrgQueryDragitem – Get DRAGITEM Structure 49
DrgQueryDragitemCount – Get Dragged Object Count 50

© Copyright IBM Corp. 1990

DrgQueryDragitemPtr — Get DRAGITEM Structure 51
DrgQueryNativeRMF — Get Format of a Dragged Object 52
DrgQueryNativeRMFLen — Get String Length of a Dragged Object 53
DrgQueryStrName — Get String Contents 54
DrgQueryStrNameLen — Get String Length 55
DrgQueryTrueType — Get True Type of Dragged Object 56
DrgQueryTrueTypeLen — Get String Length of Dragged Object 57
DrgReleasePS — Release Presentation Space 58
DrgSendTransferMsg — Send Drag Message 59
DrgSetDragImage — Set Drag Image 60
DrgSetDragitem — Set Values in DRAGITEM 61
DrgSetDragPointer — Set Mouse Pointer 62
DrgVerifyNativeRMF — Verify Rendering Mechanism and Format 63
DrgVerifyRMF — Verify Rendering Mechanism and Format 64
DrgVerifyTrueType — Verify True Type of Dragged Object 65
DrgVerifyType — Verify Type of Dragged Object 66
DrgVerifyTypeSet — Verify Types 67

Chapter 4. Graphics Function Calls 69
GpiAssociate — Associate 69
GpiCharStringPos — Character String Position 69
GpiCharStringPosAt — Character String Position At 70
GpiCreateLogColorTable — Create Logical Color Table 70
GpiCreateLogFont — Create Logical Font 70
GpiDeleteBitmap — Delete Bit Map 70
GpiPlayMetafile — Play Metafile 71
GpiQueryBitmapParameters — Query Bit-Map Parameters 71
GpiQueryCharStringPos — Query Character String Positions 71
GpiQueryCharStringPosAt — Query Character String Positions At 71
GpiQueryDefCharBox — Query Default Graphics Character Box 71
GpiQueryLogColorTable — Query Logical Color Table 71
GpiQueryTextBox — Query Text Box 72
GpiRealizeColorTable — Realize Color Table 72
GpiResetPS — Reset Presentation Space 72
GpiSetCharBox — Set Character Box 72
GpiSetClipRegion — Set Clip Region 73
GpiSetMarkerSet — Set Marker Set 73
GpiSetMetaFileBits — Set Metafile Bits 73
GpiSetPageViewport — Set Page Viewport 73
GpiSetPatternSet — Set Pattern Set 73
GpiUnrealizeColorTable — Unrealize Color Table 73

Chapter 5. Picture Function Calls 75
PicIchg — Picture Interchange Convert 75
PicPrint — Picture Print 75

Chapter 6. Profile Function Calls 77
PrfQueryProgramTitles — Query Program Titles 77
PrfReset — Reset Presentation Manager 77

Chapter 7. Spooler Function Calls 79
DosPrintDestAdd — Add Print Destination 80
DosPrintDestControl — Control Print Destination 82
DosPrintDestDel — Delete Print Destination 83
DosPrintDestEnum — Enumerate Print Destination 84
DosPrintDestGetInfo — Get Print Destination Information 85

DosPrintDestSetInfo — Set Print Destination Information 87
DosPrintJobContinue — Continue Print Job 89
DosPrintJobDel — Delete Print Job 90
DosPrintJobEnum — Enumerate Print Job 91
DosPrintJobGetId — Get Print Job ID 92
DosPrintJobGetInfo — Get Print Job Information 93
DosPrintJobPause — Pause Print Job 94
DosPrintJobSetInfo — Set Print Job Information 95
DosPrintQAdd — Add Print Queue 97
DosPrintQContinue — Continue Print Queue 99
DosPrintQDel — Delete Print Queue 100
DosPrintQEnum — Enumerate Print Queue 101
DosPrintQGetInfo — Get Print Queue Information 103
DosPrintQPause — Pause Print Queue 105
DosPrintQPurge — Purge Print Queue 106
DosPrintQSetInfo — Set Print Queue Information 107

Chapter 8. Advanced Video Function Calls 109
VioQueryFonts — Vio Query Fonts 109
VioSetDeviceCellSize — Vio Set Device Cell Size 109

Chapter 9. Window Function Calls 111
Message Queues and Window Calls 112
WinAddSwitchEntry — Add Switch Entry 114
WinAllocMem — Allocate Heap Space 114
WinBeginPaint — Begin Paint 114
WinCalcFrameRect — Calculate Frame Rectangle 114
WinChangeSwitchEntry — Change Switch Entry 114
WinCopyRect — Copy Rectangle 115
WinCreateCursor — Create Cursor 115
WinCreateHeap — Create Heap 115
WinCreateMsgQueue — Create Message Queue 115
WinCreateStdWindow — Create Standard Window 115
WinCreateSwitchEntry — Create Switch Entry 115
WinDestroyWindow — Destroy Window 116
WinDrawBitmap — Draw Bit Map 116
WinDrawBorder — Draw Border 116
WinDrawText — Draw Text 116
WinEnumDlgItem — Enumerate Dialog Item 116
WinEqualRect — Equal Rectangle 116
WinFillRect — Fill Rectangle 117
WinFreeMem — Free Memory on Heap 117
WinGetErrorInfo — Get Error Information 117
WinInflateRect — Inflate Rectangle 117
WinInitialize — Initialize 117
WinIntersectRect — Intersect Rectangle 117
WinInvalidateRect — Invalidate Rectangle 118
WinInvertRect — Invert Rectangle 118
WinIsRectEmpty — Is Rectangle Empty 118
WinLoadDlg — Load Dialog 118
WinLockVisRegions — Lock Visible Regions 118
WinMessageBox — Message Box 118
WinOffsetRect — Offset Rectangle 119
WinPtInRect — Point In Rectangle 119
WinQueryQueueStatus — Query Queue Status 119
WinQuerySysValue — Query System Value 119

WinQueryUpdateRect — Query Update Rectangle . . . 119
WinQueryWindowRect — Query Window Rectangle . . . 120
WinQueryWindowUshort — Query Window Short . . . 120
WinReallocMem — Reallocate Memory In Heap . . . 120
WinRegisterClass — Register Window Class . . . 120
WinScrollWindow — Scroll Window . . . 120
WinSetClassMsgInterest — Set Class Message Interest . . . 120
WinSetClipbrdData — Set Clipboard Data . . . 121
WinSetHook — Set Hook . . . 121
WinSetMsgInterest — Set Message Interest . . . 122
WinSetRect — Set Rectangle . . . 122
WinSetRectEmpty — Set Rectangle Empty . . . 122
WinSetSysValue — Set System Value . . . 122
WinSubtractRect — Subtract Rectangle . . . 122
WinTrackRect — Draw Tracking Rectangle . . . 123
WinUnionRect — Union Rectangle . . . 125
WinValidateRect — Validate Rectangle . . . 125

Volume 2: (Related Information) . . . 127

Chapter 10. Functions Supplied by Applications . . . 129
JournalPlaybackHook . . . 129
MsgFilterHook . . . 129

Chapter 12. Default Window Procedure Message Processing . . . 131
WM_HITTEST . . . 131
WM_SETSELECTION . . . 131

Chapter 13. Button Control Window Processing . . . 133
Button Control Styles . . . 133

Chapter 14. Entry Field Control Window Processing . . . 135
Entry Field Control Styles . . . 135

Chapter 15. Frame Control Window Processing . . . 137
WM_FORMATFRAME . . . 137

Chapter 16. List Box Control Window Processing . . . 139
List Box Control Styles . . . 139
WM_DRAWITEM . . . 139
WM_MEASUREITEM . . . 139
LM_QUERYCURSOR . . . 139
LM_SETCURSOR . . . 139
LM_SETSELECTION . . . 140

Chapter 17. Menu Control Window Processing . . . 141
MM_DISMISSMENU . . . 141
MM_QUERYITEMTEXT . . . 141
MM_SELECTITEM . . . 141

Chapter 18. Multi-Line Entry Field Control Window Processing . . . 143
WM_CONTROL . . . 143
MLM_FORMAT . . . 143
MLM_QUERYSEL . . . 143
MLM_SEARCH . . . 144

MLM_SETFORMATRECT 144

Chapter 19. Prompted Entry Field Control Window Processing 145
WM_CONTROL 145
Combo Box Control Window Messages 145
WM_UPDATESTYLE 145

Chapter 20. Scroll Bar Control Window Processing 147
SBM_QUERYHILITE 147
SBM_SETHILITE 147
SBM_SETSCROLLBAR 147

Chapter 20.1. Spin Button Control Window Processing 149
Types of Spin Button Control 149
SPBM_OVERRIDESETLIMITS 150
SPBM_QUERYLIMITS 151
SPBM_QUERYVALUE 152
SPBM_SETARRAY 153
SPBM_SETCURRENTVALUE 154
SPBM_SETLIMITS 155
SPBM_SETMASTER 156
SPBM_SETTEXTLIMIT 157
SPBM_SPINDOWN 158
SPBM_SPINUP 158
WM_CONTROL (in Spin Controls) 159

Chapter 23.1. Direct Manipulation Messages 161
DM_DRAGERROR 161
DM_DRAGFILECOMPLETE 162
DM_DRAGLEAVE 163
DM_DRAGOVER 163
DM_DROP 165
DM_DROPHELP 166
DM_EMPHASIZETARGET 166
DM_ENDCONVERSATION 167
DM_FILERENDERED 168
DM_PRINT 169
DM_RENDER 169
DM_RENDERCOMPLETE 170
DM_RENDERFILE 171
DM_RENDERPREPARE 172

Chapter 24. Dynamic Data Exchange Messages 175
WM_DDE_INITIATE 175
WM_DDE_INITIATEACK 175

Chapter 25. Help Manager Messages 177
HM_ACTION_BAR_COMMAND 177

Chapter 26. Resource Files 179
Directives 179
Menu Template 179
Predefined Control Statements 179

Chapter 28. Code Pages 181
Full Screen VIO Applications 181

Font Layout 182

Appendix B. Standard Bit-Map Formats 189
Bit-Map File Format 189

Appendix D. Format of Interchange Files 193
Metafile Restrictions 193

Master Index 195

Index 197

Volume 1: (Data Types and Function Calls)

© Copyright IBM Corp. 1990

Chapter 2. Data Types

Summary of Changes

New	Updated	Section Title
	√	BITMAPINFO
	√	BITMAPINFOHEADER
√		DRAGIMAGE
√		DRAGINFO
√		DRAGITEM
√		DRAGTRANSFER
√		ESCSETMODE
	√	FONTMETRICS
√		HFILE
√		HSTR
	√	MARGSTRUCT
	√	MLE_SEARCHDATA
	√	MT
	√	OVERFLOW
√		PAPSZ
√		PRDINFO
√		PRDINFO3
√		PRIDINFO
√		PRJINFO
√		PRJINFO2
√		PRJINFO3
√		PRQINFO
√		PRQINFO3
√		RENDERFILE
√		SPLERR

BITMAPINFO

On page 2-2, change the description of the **length** field to:

length
Length of structure.

This value depends on the particular language in use.

© Copyright IBM Corp. 1990

BITMAPINFOHEADER

On page 2-3, change the description of the **length** field to:

length
Length of structure.

This value depends on the particular language in use.

DRAGIMAGE

Dragged-object-image structure.

cb *(USHORT)*
Size, in bytes, of the *DRAGIMAGE* structure.

cptl *(USHORT)*
The number of points in the point array if **fl** is specified as DRG_POLYGON.

hImage *(HANDLE)*
Handle representing the image to display.

The type is determined by **fl**.

stretch *(SIZEL)*
Dimensions for stretching.

Specifies the dimensions for stretching when **fl** is specified as DRG_STRETCH.

fl *(ULONG)*
Flags:

DRG_ICON **hImage** is a *HPOINTER*.
DRG_BITMAP **hImage** is a *HBITMAP*.
DRG_POLYGON
hImage is a pointer to an array of points that will be connected with GpiPolyLine to form a polygon. The first point of the array should be (0,0), and the other points should be placed relative to this position.
DRG_STRETCH
If DRG_ICON or DRG_BITMAP is specified, the image will be expanded or compressed to the dimensions specified by **stretch**.
DRG_TRANSPARENT
If DRG_ICON is specified, an outline of the icon will be generated and displayed instead of the original icon.
DRG_CLOSED If DRG_POLYGON is specified, a closed polygon will be formed by moving the current position to the last point in the array before calling GpiPolyLine.

xoffset *(SHORT)*
x offset from the pointer hot spot to the origin of the image.

yoffset *(SHORT)*
y offset from the pointer hot spot to the origin of the image.

DRAGINFO

Drag-information structure.

dragInfo *(COUNT4B)*
Size, in bytes, of the structure.

The size includes the array of *DRAGITEM* structures.

dragItem *(COUNT2B)*
Size, in bytes, of each *DRAGITEM* structure.

operation *(SHORT)*
Modified drag operations.

An application can define its own modified drag operations for use when simulating a drop. These operations must have a value greater than DO_UNKNOWN.

DO_DEFAULT	Execute the default drag operation. No modifier keys are pressed.
DO_COPY	Execute a copy operation. The Ctrl key is pressed.
DO_MOVE	Execute a move operation. The Alt key is pressed.
DO_UNKNOWN	An undefined combination of modifier keys is pressed.

source *(HWND)*
Window handle of the source of the drag operation.

xdrop *(SHORT)*
x coordinate of drop point expressed in desktop coordinates.

ydrop *(SHORT)*
y coordinate of drop point expressed in desktop coordinates.

cItem *(COUNT2)*
Count of *DRAGITEM* structures.

reserved *(USHORT)*
Reserved.

DRAGITEM

Drag-object structure.

hItem *(HWND)*
Window handle of the source of the drag operation.

ulItemId *(ULONG)*
Information used by the source to identify the object being dragged.

type *(HSTR)*
String handle of the object type.

The string handle must be created using the DrgAddStrHandle call. The string is of the form: TYPE[,TYPE...] . The first type in the list must be the true type of the object.

The following types are used by OS/2:

DRT_ASM	Assembler code
DRT_BASIC	BASIC code
DRT_BINDATA	Binary data
DRT_BITMAP	Bit map

DRT_C	C code
DRT_COBOL	COBOL code
DRT_DLL	Dynamic link library
DRT_DOSCMD	DOS command file
DRT_EXE	Executable file
DRT_FONT	Font
DRT_FORTRAN	FORTRAN code
DRT_ICON	Icon
DRT_LIB	Library
DRT_METAFILE	Metafile
DRT_OS2CMD	OS/2 command file
DRT_PASCAL	Pascal code
DRT_RESOURCE	Resource file
DRT_TEXT	Text
DRT_UNKNOWN	Unknown type.

rmf *(HSTR)*
String handle of the rendering mechanism and format.

The string handle must be created using the DrgAddStrHandle call. The string is of the form: MECHFMT[,MECHFMT...] , where MECHFMT can be in either of the following formats:

1. <mechanism(1),format(1)>
2. (mechanism(1)[, mechanism(n)...]) x (format(1)[,format(n)...])

The first mechanism/format pair must be the native rendering mechanism and format of the object.

Valid mechanisms are:

DRM_DDE	Dynamic data exchange
DRM_OS2FILE	OS/2 file
DRM_PRINT	Object can be printed using direct manipulation.

Valid formats are:

DRF_BITMAP	OS/2 bit map
DRF_DIB	DIB
DRF_DIF	DIF
DRF_DSPBITMAP	Stream of bit-map bits
DRF_METAFILE	Metafile
DRF_OEMTEXT	OEM text
DRF_OWNERDISPLAY	Bit stream
DRF_PTRPICT	Printer picture
DRF_RTF	Rich text
DRF_SYLK	SYLK
DRF_TEXT	Null-terminated string
DRF_TIFF	TIFF
DRF_UNKNOWN	Unknown format.

containername *(HSTR)*
String handle of the name of the container holding the source object.

The string handle must be created using the DrgAddStrHandle call.

sourcename *(HSTR)*
String handle of the name of the source object.

The string handle must be created using the DrgAddStrHandle call.

targetname *(HSTR)*
String handle of the suggested name of the object at the target.

It is the responsibility of the source of the drag operation to create this string handle before calling DrgDrag.

xoffset *(SHORT)*
x offset from the pointer hot spot to the origin of the image that represents this object.

This value is copied from **xoffset** in the *DRAGIMAGE* structure by DrgDrag.

yoffset *(SHORT)*
y offset from the pointer hot spot to the origin of the image that represents this object.

This value is copied from **yoffset** in the *DRAGIMAGE* structure by DrgDrag.

control *(BIT16)*
Source-object control flags:

DC_OPEN	Object is open
DC_REF	Reference to another object
DC_GROUP	Group of objects
DC_CONTAINER	Container of other objects
DC_PREPARE	Source requires a DM_RENDERPREPARE message before it establishes a data transfer conversation
DC_REMOVEABLEMEDIA	Object is on removeable media, or object cannot be recovered after a move operation.

supportedops *(BIT16)*
Direct manipulation operations supported by the source object:

DO_COPYABLE	Source supports DO_COPY
DO_MOVEABLE	Source supports DO_MOVE.

DRAGTRANSFER

Drag-conversation structure.

transferinfo *(COUNT4B)*
Size, in bytes, of the structure.

client *(HWND)*
Handle of the client window.

This can be the target window or a window that represents an object in a container that was dropped on.

ditem *(DRAGITEM)*
The *DRAGITEM* structure that is to be rendered.

This structure must exist within the *DRAGINFO* structure that was passed in the DM_DROP message.

selectedrmf *(HSTR)*
The string handle for the selected rendering mechanism and format for the transfer operation.

This handle must be created using DrgAddStrHandle. The target is responsible for deleting this handle when the conversation is complete. The string is in the format: <MECHANISM,FORMAT>.

rendertoname *(HSTR)*
A string handle representing the name where the source will place, and the target will find, the data that is rendered.

The target is responsible for deleting this string handle when the conversation terminates. The contents of this field vary according to the rendering mechanism. See **rmf** in *DRAGITEM*.

OS/2 File	The string handle represents the fully qualified name of the file where the rendering will be placed.
DDE	This field is not used.
Print	This field is not used.

targetinfo *(ULONG)*
Reserved for use by the target.

The target can use this field for information about the object and rendering operation.

operation *(BIT16)*
The operation.

Values are:

DO_COPY	Execute a copy operation.
DO_MOVE	Execute a move operation.
OTHER	Execute an application-defined operation.

reply *(BIT16)*
Reply flags for the message.

These flags can be set as follows:

DMFL_NATIVERENDER	The source does not support rendering for this object. A source should not set this flag unless it provides sufficient information at the time of the drop for the target to perform the rendering operation. The target must send DM_ENDCONVERSATION to the source after carrying out the rendering operation, or when it elects not to do a native rendering.
DMFL_RENDERRETRY	The source supports rendering for the object, but does not support the selected rendering mechanism and format. The target can try another mechanism and format by sending another DM_RENDER message. If the target does not retry, it must send a DM_RENDERCOMPLETE message to the source. This flag is set in conjunction with the DMFL_NATIVERENDER flag.

ESCSETMODE

Structure for setting printer mode. See DevEscape (DEVESC_SETMODE) on page 29 in this book.

mode *(ULONG)*
Mode

Mode to be set.

0 Set mode to specified code page. Any font can be used.

codepage *(USHORT)*
Code page.

If zero is specified for the code page, the printer is set to the hardware default.

FONTMETRICS

On page 2-11, this data type should be described as follows.

FONTMETRICS Font-metrics structure.

familyname *(CHAR*FACESIZE)*
Family name.

The family name of the font that describes the basic appearance of the font, for example, Times New Roman.

facename *(CHAR*FACESIZE)*
Facename.

The typeface name that defines the particular font, for example, Times New Roman Bold Italic.

registry *(IDENTITY)*
Registry identifier.

The IBM registered number (or zero).

codepage *(CPID)*
Code page.

Defines the registered code page supported by the font. For example, the original IBM PC code page is 437. A value of 0 implies that the font may be used with any of the OS/2 supported code pages.

Where a font contains special symbols for which there is no registered code page, then code page 65400 is used.

emheight *(LONG)*
Em height.

The height of the Em square in world coordinate units. This corresponds to the point size for the font.

xheight *(LONG)*
x height.

The nominal height above the baseline for lowercase characters (ignoring ascenders) in world coordinate units.

maxascender *(LONG)*
Maximum ascender.

The maximum height above the baseline reached by any part of any symbol in the font in world coordinate units.

maxdescender *(LONG)*
Maximum descender.

The maximum depth below the baseline reached by any part of any symbol in the font in world coordinate units.

lowercaseascent *(LONG)*
Lowercase ascent.

The maximum height above the baseline reached by any part of any lowercase (Latin 'a' through 'z') symbol in the font in world coordinate units.

lowercasedescent *(LONG)*
Lowercase descent.

The maximum depth below the baseline reached by any part of any lowercase (Latin 'a' through 'z') symbol in the font in world coordinate units.

internalleading *(LONG)*
Internal leading.

The amount of white space normally separating lines of text where lines are spaced by the maximum baseline extent (world coordinate units). This space may contain accents.

externalleading *(LONG)*
External leading.

The white space, in addition to the internal leading, to separate lines of text (world coordinate units). This value may be zero.

avecharwidth *(WIDTH4)*
Average character width.

Weighted average inter-character increment for the font in world coordinate units.

maxcharinc *(WIDTH4)*
Maximum character increment.

The maximum character increment for the font in world coordinate units.

eminc *(WIDTH4)*
Em increment.

The width of the Em square in world coordinate units. This corresponds to the point size of the font. When the horizontal device resolution equals the vertical device resolution this is equal to the emheight.

maxbaselineext *(WIDTH4)*
Maximum baseline extent.

The maximum vertical space occupied by the font, in world coordinate units. This is the sum of **maxascender** and **maxdescender** if both are positive. It is also the sum of **internalleading** and **emheight**.

A possible line spacing can be computed by adding **maxbaselineext** to **externalleading** although more sophisticated applications will base line spacing on the point size (**emheight**) of the font (say 120%) plus any external leading.

charslope *(SHORT)*
Character slope.

Defines the nominal slope for the characters of a font. The

slope is defined in degrees increasing clockwise from the vertical. An *Italic* font is an example of a font with a nonzero slope.

Programming Note: This angle is contained in a two-part unsigned number. The first 9 bits contain a value in the range 0 through 359, representing the number of degrees in the rotation. The next 6 bits contain a number in the range 0 through 59, representing the number of minutes in the rotation. The final bit is reserved 0. Values outside the specified ranges are not valid.

inlinedir *(SHORT)*
Inline direction.

The direction in which the characters in the font are designed for viewing, in degrees increasing clockwise from the horizontal (left-to-right). Characters are added to a line of text in the inline direction.

Programming Note: This angle is contained in a two-part unsigned number. The first 9 bits contain a value in the range 0 through 359, representing the number of degrees in the rotation. The next 6 bits contain a number in the range 0 through 59, representing the number of minutes in the rotation. The final bit is reserved 0. Values outside the specified ranges are not valid.

charrot *(SHORT)*
Character rotation.

The rotation of the character glyphs with respect to the baseline.

Programming Note: This angle is contained in a two-part unsigned number. The first 9 bits contain a value in the range 0 through 359, representing the number of degrees in the rotation. The next 6 bits contain a number in the range 0 through 59, representing the number of minutes in the rotation. The final bit is reserved 0. Values outside the specified ranges are not valid.

weightclass *(PROPERTY2)*
Weight class.

Indicates the visual weight (thickness of strokes) of the characters in the font:

Value	Description
1	Ultra-light
2	Extra-light
3	Light
4	Semi-light
5	Medium (normal)
6	Semi-bold
7	Bold
8	Extra-bold
9	Ultra-bold

widthclass *(PROPERTY2)*
Width class.

Indicates the relative aspect ratio of the characters of the font in relation to the normal aspect ratio for this type of font:

Value	Description	% of normal width
1	Ultra-condensed	50
2	Extra-condensed	62.5
3	Condensed	75
4	Semi-condensed	87.5
5	Medium (normal)	100
6	Semi-expanded	112.5
7	Expanded	125
8	Extra-expanded	150
9	Ultra-expanded	200

xdeviceres *(SHORT)*
x device resolution.

The x-resolution of the target device. For raster fonts the units are pels per inch and for outline fonts the units are the logical units defining the drawing grid.

ydeviceres *(SHORT)*
y device resolution.

The y-resolution of the target device. For raster fonts the units are pels per inch and for outline fonts the units are the logical units defining the drawing grid.

firstchar *(SHORT)*
First character.

The code point of the first character in the font.

lastchar *(SHORT)*
Last character.

The code point of the last character in the font, expressed as an offset from **firstchar**.

All code points between the first and last character specified must be supported by the font.

defaultchar *(SHORT)*
Default character.

The code point that is used if a code point outside the range supported by the font is used, expressed as an offset from **firstchar**.

breakchar *(SHORT)*
Break character.

The code point that represents the 'space' or 'break' character for this font, expressed as an offset from **firstchar**. For example, if the first character is the space in code page 850, **firstchar** = 32, and **breakchar** = 0.

nominalpointsize *(SHORT)*
Nominal point size.

The height of the font specified in *decipoints* (1/720 inch). The nominal size is the size for which the font is designed.

minimumpointsize *(SHORT)*
Minimum point size.

The minimum height in decipoints to which the font can be scaled down for display while retaining fidelity.

maximumpointsize *(SHORT)*
Maximum point size.

The maximum height in decipoints to which the font can be scaled up for display while retaining fidelity.

type *(BIT16)*
Type indicators.

Contains this information:

FM_TYPE_FIXED
Characters in the font have the same fixed width.
FM_TYPE_LICENSED
Licensed (protected) font.
FM_TYPE_KERNING
Font contains kerning information.
FM_TYPE_64K
Font is larger than 64KB (KB equals 1024 bytes) in size. If the following two bits are false, the font is for single-byte code pages. One of the bits may be set.
FM_TYPE_DBCS
Font is for double-byte code pages.
FM_TYPE_MBCS
Font is for mixed single/double-byte code pages.

defn *(BIT16)*
Definition indicators.

Contains the following font definition data:

FM_DEFN_OUTLINE
Font is a vector (outline) font, otherwise it is a bit-map font.
FM_DEFN_GENERIC
Font is in a format that can be used by the GPI, otherwise it is a device font.

selection *(BIT16)*
Selection indicators.

Contains information about the font patterns in the physical font.

Note: The flags do not reflect simulations applied to the physical font.

FM_SEL_ITALIC
Font contains Italic characters, otherwise they are upright.
FM_SEL_UNDERSCORE
Characters are underscored.

FM_SEL_NEGATIVE
Characters have their foreground and background reversed.
FM_SEL_OUTLINE
Outline (hollow) characters, otherwise they are solid.
FM_SEL_STRIKEOUT
Characters are overstruck.
FM_SEL_BOLD
Characters are bold.

capabilities *(BIT16)*
Capabilities.

This tells the engine the kind of operations it can perform on the font.

FM_CAP_NOMIX
Characters may not be mixed with graphics.
QUALITY
The most significant byte may contain the following numeric value:

0	Undefined
1	DP quality
2	DP draft
3	Near Letter Quality
4	Letter Quality

This attribute applies only to device fonts.

subscriptxsize *(LONG)*
Subscript x size.

The recommended horizontal size for subscripts for this font in world coordinate units.

subscriptysize *(LONG)*
Subscript y size.

The recommended vertical size for subscripts for this font in world coordinate units.

subscriptxoffset *(LONG)*
Subscript x offset.

The recommended baseline x-offset for subscripts for this font in world coordinate units.

subscriptyoffset *(LONG)*
Subscript y offset.

The recommended baseline y-offset for subscripts for this font in world coordinate units.

superscriptxsize *(LONG)*
Superscript x size.

The recommended horizontal size for superscripts for this font in world coordinate units.

superscriptysize *(LONG)*
Superscript y size.

The recommended vertical point size for superscripts for this font in world coordinate units.

superscriptxoffset *(LONG)*
Superscript x offset.

The recommended baseline x-offset for superscripts for this font in world coordinate units.

superscriptyoffset *(LONG)*
Superscript y offset.

The recommended baseline y-offset for superscripts for this font in world coordinate units.

underscoresize *(LONG)*
Underscore size.

The width of the underscore stroke in world coordinate units.

underscoreposition *(LONG)*
Underscore position.

The position of the underscore stroke from the baseline in world coordinate units.

strikeoutsize *(LONG)*
Strikeout size.

The width of the strikeout stroke in world coordinate units.

strikeoutposition *(LONG)*
Strikeout position.

The position of the strikeout stroke relative to the baseline in world coordinate units.

kerningpairs *(SHORT)*
Kerning pairs.

The number of kerning pairs in the kerning pair table.

familyclass *(SHORT)*
Font family design classification.

This value contains a font class and its subclass.

match *(PROPERTY4)*
Matched font identity.

This uniquely identifies the font for a given device/device driver combination. A positive match number signifies that the font is a generic (engine) font while a negative number indicates a device font (a native or downloadable font).

HFILE

Resource handle.

HSTR

String handle.

MARGSTRUCT

On page 2-21, this data type is now named *MLEMARGSTRUCT*.

Change the constants listed under the **flags** parameter as follows:

MLF_MARGINLEFT should be MLFMARGIN_LEFT
MLF_MARGINRIGHT should be MLFMARGIN_RIGHT
MLF_MARGINTOP should be MLFMARGIN_TOP
MLF_MARGINBOTTOM should be MLFMARGIN_BOTTOM.

MLE_SEARCHDATA

On page 2-23, this data type should read:

size *(COUNT2B)*
Size of *MLE_SEARCHDATA* structure.

findstring *(STR)*
String to search for.

replacestring *(STR)*
String to replace with.

findstringlength *(SHORT)*
Length of **findstring** string.

replacestringlength *(SHORT)*
Length of **replacestring** string.

start *(IPT)*
Point at which to start search, or point where string was found.

non-negative Point at which to start search
negative Start search from current cursor location.

stop *(IPT)*
Point at which to stop search.

non-negative Point at which to stop search
negative Stop search at end of text.

foundstringlength *(SHORT)*
Length of string found at **start**.

MT

On page 2-24, this data type should be described as follows.

length *(COUNT2B)*
Length of menu template in bytes.

version *(USHORT)*
Template version.

Versions 0 and 1 are valid.

codepage *(USHORT)*
Code page.

itemoffset *(USHORT)*
Offset of items from start of template.

count *(COUNT2)*
Count of menu items.

presparamsoffset *(USHORT)*
Offset of presentation parameters from start of template.

This field does not exist for version 0 of the template.

menuItems *(MTI*count)*
Menu items.

OVERFLOW

On page 2-24, this data type is now named *MLEOVERFLOW*.

PAPSZ

Pointer to an array of pointers to null-terminated strings.

PRDINFO

Print-destination information structure (level 1).

name *(CHAR*PDLEN+1)*
Name of logical address.

This is the logical device address; for example, LPT1.

username *(CHAR*UNLEN+1)*
User who submitted job.

This parameter is valid only while the job is printing.

jobId *(IDENTITY)*
Identity of current job.

If 0, no job is printing.

status *(BIT16)*
Print destination status.

Use the mask PRD_STATUS_MASK to determine the print job status:

PRD_ACTIVE	Processing
PRD_PAUSED	Not processing, or paused.

Use the mask PRJ_DEVSTATUS for further information about print job status:

PRJ_COMPLETE	Job complete
PRJ_INTERV	Intervention required
PRJ_ERROR	Error occurred (in this case, **statuscomment** may contain a comment about the error)
PRJ_DESTOFFLINE	Print destination offline
PRJ_DESTPAUSED	Print destination paused
PRJ_NOTIFY	Raise alert
PRJ_DESTNOPAPER	Print destination out of paper.

statuscomment *(STRL)*
Print destination comment while printing.

A comment posted by the queue processor of the print destination. This parameter is valid only during printing.

time *(USHORT)*
Time job has been printing (minutes).

This parameter applies only during printing.

PRDINFO3

Print-destination information structure (level 3).

printername *(STRL)*
Printer name.

username *(STRL)*
User who submitted job.

This parameter is valid only while the job is printing. It is NULL for a job submitted locally.

logaddr *(STRL)*
Logical address.

If NULL, the printer is not connected to a logical address.

jobid *(IDENTITY)*
Identity of current job.

If 0, no job is printing.

status *(BIT16)*
Print destination status.

Use the mask PRD_STATUS_MASK to determine the print job status:

PRD_ACTIVE	Processing
PRD_PAUSED	Not processing, or paused.

Use the mask PRJ_DEVSTATUS for further information about print job status:

PRJ_COMPLETE	Job complete
PRJ_INTERV	Intervention required
PRJ_ERROR	Error occurred (in this case, **statuscomment** may contain a comment about the error)
PRJ_DESTOFFLINE	Print destination offline
PRJ_DESTPAUSED	Print destination paused
PRJ_NOTIFY	Raise alert
PRJ_DESTNOPAPER	Print destination out of paper.

statuscomment *(STRL)*
Print destination comment while printing.

A comment posted by the print processor of the print destination. This parameter is valid only during printing.

comment *(STRL)*
Printer description.

drivers *(STRL)*
Drivers supported by printer.

List items are separated by commas. Each driver name may have a device name separated by a dot (for example, PLOTTER.HP7475A).

time *(USHORT)*
Time job has been printing (minutes).

This parameter applies only during printing.

timeout *(USHORT)*
Device timeout (seconds).

The time that elapses before the device driver notifies the spooler that the device has not responded.

PRIDINFO

Print-job identification-number information structure.

jobid *(IDENTITY)*
Print-job identification number.

server *(CHAR*CNLEN+1)*
Server.

The name of the remote server handling the print job.

qname *(CHAR*QNLEN+1)*
Queue name.

The name of the queue the job is on.

pad_1 *(CHAR)*
Pad character.

PRJINFO

Print-job information structure (level 1).

jobid *(IDENTITY)*
Job identification number.

username *(CHAR*UNLEN+1)*
User who submitted the job.

This parameter applies only to jobs created by a user and queued on a remote server. A NULL string signifies a local job.

pad_1 *(CHAR)*
Pad character.

notifyname *(CHAR*CNLEN+1)*
Messaging alias for print alert.

This parameter applies only to jobs queued on a remote server. A NULL string is returned for a local job.

datatype *(CHAR*DTLEN+1)*
Data type of submitted file.

This is specified by the **datatype** parameter in the *DEVOPENSTRUC* as passed to the DevOpenDC or SplQmOpen call when the job is created. The name is truncated to fit the field if necessary, and contains a trailing NULL.

parms *(STRL)*
Parameters.

The form of this string is:

```
parm1=val1 parm2=val2 ...
```

position *(USHORT)*
Job position in queue.

If 1, the job is scheduled to be the next job printed from this queue.

status *(BIT16)*
Job status.

To find the job status, use the PRJ_QSTATUS mask:

PRJ_QS_QUEUED	Queued
PRJ_QS_PAUSED	Paused by a DosPrintJobPause call
PRJ_QS_SPOOLING	Job being created
PRJ_QS_PRINTING	Printing (bits 2 through 8 are valid).

If the job status is PRJ_QS_PRINTING, use the PRJ_DEVSTATUS mask to find the device status. The 1-bit flags used for device status are:

PRJ_COMPLETE	Job complete
PRJ_INTERV	Intervention required
PRJ_ERROR	Error occurred (**statuscomment** may contain additional information)
PRJ_DESTOFFLINE	Print destination offline
PRJ_DESTPAUSED	Print destination paused
PRJ_NOTIFY	Alert should be raised
PRJ_DESTNOPAPER	Print destination out of paper.

This bit indicates that the job is deleted:

PRJ_DELETED	Job deleted.

statuscomment *(STRL)*
Status comment.

A text string, posted by the print-queue processor, that provides additional job-status information. The default string type is NULL.

submitted *(ULONG)*
Time job submitted.

Time format is the same as that stored in the global information segment.

size *(COUNT4B)*
Print-job size (bytes).

comment *(STRL)*
Comment string about print job.

The maximum length of the string is 47 characters.

PRJINFO2

Print-job information structure.

This structure provides a subset of the information supplied by *PRJINFO3*. It minimizes the storage required for job-information retrieval, and is sufficient for most uses.

jobId *(IDENTITY)*
Job identification number.

priority *(USHORT)*
Job priority.

The job-priority range is 1 through 99, with 99 the highest job priority. (For queue priorities, 1 is the highest priority.)

The job priority determines the order of jobs in the queue. If multiple queues print to the same printer, the job at the front of each queue is examined. The job with the highest priority is printed first; if there is more than one job with the highest priority, the oldest job with this priority is printed first.

PRJ_MAX_PRIORITY	Highest priority
PRJ_MIN_PRIORITY	Lowest priority
PRJ_NO_PRIORITY	No priority.

username *(STRL)*
User who submitted the job.

This parameter applies only to jobs created by a user and enqueued on a remote server. A NULL string signifies a local job.

position *(USHORT)*
Job position in queue.

If 1, the job is scheduled to be the next job printed from this queue.

status *(BIT16)*
Job status.

To find the job status, use the PRJ_QSTATUS mask:

PRJ_QS_QUEUED	Queued
PRJ_QS_PAUSED	Paused by a DosPrintJobPause call
PRJ_QS_SPOOLING	Job being created
PRJ_QS_PRINTING	Printing (bits 2 through 11 are valid).

For further information, use the PRJ_DEVSTATUS mask:

PRJ_COMPLETE	Job complete
PRJ_INTERV	Intervention required
PRJ_ERROR	Error occurred (**comment** may contain additional information)
PRJ_DESTOFFLINE	Print destination offline
PRJ_DESTPAUSED	Print destination paused
PRJ_NOTIFY	Alert should be raised
PRJ_DESTNOPAPER	Print destination out of paper
PRJ_DESTFORMCHG	Printer waiting for form change
PRJ_DESTCRTCHG	Printer waiting for cartridge change
PRJ_DESTPENCHG	Printer waiting for pen change.

This bit indicates that the job is deleted:

PRJ_DELETED	Job deleted.

submitted *(ULONG)*
Time job submitted.

Time format is the same as that stored in the global information segment.

size *(COUNT4B)*
Print-job size (bytes).

comment *(STRL)*
Comment string.

Information about the print job. The maximum length of the string is 47 characters.

document *(STRL)*
Document name.

The document name of the print job (set by the application that submitted the print job). The maximum length of the string is 260 characters.

PRJINFO3

Print-job information structure.

This structure is used when complete job details are required. A subset of this information is supplied by *PRJINFO2*.

jobId *(IDENTITY)*
Job identification number.

priority *(USHORT)*
Job priority.

The job-priority range is 1 through 99, with 99 the highest job priority. (For queue priorities, 1 is the highest priority.)

The job priority determines the order of jobs in the queue. If multiple queues print to the same printer, the job on the front of each queue is examined. The job with the highest priority is printed first; if there is more than one job with the highest priority, the oldest job with this priority is printed first.

PRJ_MAX_PRIORITY	Highest priority
PRJ_MIN_PRIORITY	Lowest priority
PRJ_NO_PRIORITY	No priority.

username *(STRL)*
User who submitted the job.

This parameter applies only to jobs created by a user on a remote workstation and queued on a server. A NULL string signifies a local job.

position *(USHORT)*
Job position in queue.

If 1, the job is scheduled to be the next job printed from this queue.

status *(BIT16)*
Job status.

To find the job status, use the PRJ_QSTATUS mask:

PRJ_QS_QUEUED	Queued
PRJ_QS_PAUSED	Paused by a DosPrintJobPause call
PRJ_QS_SPOOLING	Job being created
PRJ_QS_PRINTING	Printing (bits 2 through 11 are valid).

For further information, use the PRJ_DEVSTATUS mask:

PRJ_COMPLETE	Job complete
PRJ_INTERV	Intervention required
PRJ_ERROR	Error occurred (**comment** may contain additional information)

PRJ_DESTOFFLINE	Print destination offline
PRJ_DESTPAUSED	Print destination paused
PRJ_NOTIFY	Alert should be raised
PRJ_DESTNOPAPER	Print destination out of paper
PRJ_DESTFORMCHG	Printer waiting for form change
PRJ_DESTCRTCHG	Printer waiting for cartridge change
PRJ_DESTPENCHG	Printer waiting for pen change.

This bit indicates that the job is deleted:

PRJ_DELETED	Job deleted.

submitted *(ULONG)*
Time job submitted.

Time format is the same as that stored in the global information segment.

size *(COUNT4B)*
Print-job size (bytes).

comment *(STRL)*
Comment string.

Information about the print job.

The maximum length of the string is 47 characters.

document *(STRL)*
Document name.

The document name of the print job (set by the application that submitted the print job). The maximum length of the string is 260 characters.

notifyname *(STRL)*
Messaging alias for print alert.

This parameter applies only to a jobs queued on a remote server. A NULL string is returned for a local job.

datatype *(STRL)*
Data type of submitted file.

This is specified by the **datatype** parameter in the *DEVOPENSTRUC* structure passed to the DevOpenDC or SplQmOpen call when the job is created. The name is truncated to fit the field if necessary, and contains a trailing NULL.

parms *(STRL)*
Parameters.

The form of this string is:

```
parm1=val1 parm2=val2 ...
```

statuscomment *(STRL)*
Status comment.

A text string, posted by the print-queue processor, that provides additional job-status information. The default string type is NULL.

queue *(STRL)*
Queue name.

The name of the queue the job is on.

qprocname *(STRL)*
Queue processor.

The name of the queue processor.

qprocparms *(STRL)*
Queue-processor parameters.

Spaces are used to separate parameters.

drivername *(STRL)*
Driver name.

The name of the device driver (for example, IBM4201). The device name is part of **driverdata**.

driverdata *(PDRIVDATA)*
Driver data.

The contents are specific to the device driver. The *DRIVDATA* structure is defined in the Operating System/2 (OS/2) Version 1.2 *Presentation Manager Programming Reference*.

printername *(STRL)*
Printer name.

If the job is printing, the printer name, otherwise NULL.

PRQINFO

Print-queue information structure for existing applications (level 1).

name *(CHAR*QNLEN+1)*
Queue name.

The maximum length of the name in the network case is 31 bytes (including one byte for zero termination).

pad_1 *(CHAR)*
Pad character.

priority *(USHORT)*
Queue priority.

The range is 1 through 9, with 1 being the highest queue priority. (For job priorities, 1 is the lowest priority).

The default job priority (DefJobPrio) is determined from:
DefJobPrio=100–(10* **priority**).

If a job is added with *PRJ_NO_PRIORITY* specified, DefJobPrio is used. If a default priority higher than the default job priority is specified, the default job priority is used. If a default priority lower than the default is specified, the specified job priority is used.

The queue priority does not directly affect scheduling.

PRQ_DEF_PRIORITY	Default priority
PRQ_MAX_PRIORITY	Highest priority
PRQ_MIN_PRIORITY	Minimum priority
PRQ_NO_PRIORITY	No priority.

starttime *(USHORT)*
Minutes after midnight when queue becomes active.

For example, the value 75 represents 1:15 a.m.

If **starttime** and **untiltime** are both 0, the print queue is always available.

untiltime *(USHORT)*
Minutes after midnight when queue ceases to be active.

For example, the value 1200 represents 8 p.m.

If **untiltime** and **starttime** are both 0, the print queue is always available.

sepfile *(STRL)*
Separator-page file.

The path and file name of a separator-page file on the target computer.

This file contains formatting information for the page or pages to be used between print jobs. A relative path name is taken as relative to the current spool directory. A NULL string indicates no separator page.

See *IBM Operating System/2 Local Area Network Server Version 1.2: Network Administrator's Guide* for information about the format of separator files.

This feature is supported only on network servers. It can be set and queried on a local workstation, but no separator pages will be printed unless the server configuration of the LAN Manager program is installed on the computer.

prproc *(STRL)*
Default queue-processor for queue.

printers *(STRL)*
Print destinations connected to queue.

parms *(STRL)*
Queue parameters.

This is defined by the application. It can be any text string or a NULL string.

comment *(STRL)*
Queue description.

An optional string up to 48 characters long. A NULL string results in no comment.

status *(BIT16)*
Queue status.

PRQ_ACTIVE	Active
PRQ_PAUSE	Paused
PRQ_PENDING	Deletion pending
PRQ_ERROR	Error.

jobs *(COUNT2)*
Number of jobs in queue.

PRQINFO3

Print-queue information structure.

This structure is used at information levels 3 and 4.

name *(STRL)*
Queue name.

The maximum length of the name in the network case is 256 (including one byte for zero termination).

priority *(USHORT)*
Queue priority.

The range is 1 through 9, with 1 being the highest queue priority.

The default job priority (DefJobPrio) is determined from:
DefJobPrio=100–(10* **priority**).

If a job is added with *PRJ_NO_PRIORITY* specified, DefJobPrio is used. If a default priority higher than the default job priority is specified, the default job priority is used. If a default priority lower than the default is specified, the specified job priority is used.

PRQ_DEF_PRIORITY	Default priority
PRQ_MAX_PRIORITY	Highest priority
PRQ_MIN_PRIORITY	Minimum priority
PRQ_NO_PRIORITY	No priority.

starttime *(USHORT)*
Minutes after midnight when queue becomes active.

For example, the value 75 represents 1:15 a.m.

If **starttime** and **untiltime** are both 0, the print queue is always available.

untiltime *(USHORT)*
Minutes after midnight when queue ceases to be active.

For example, the value 1200 represents 8 p.m.

If **untiltime** and **starttime** are both 0, the print queue is always available.

pad_1 *(USHORT)*
Pad character.

sepfile *(STRL)*
Separator-page file.

The path and file name of a separator-page file on the target computer.

This file contains formatting information for the page or pages to be used between print jobs. A relative path name is taken as relative to the current spool directory. A NULL string indicates no separator page.

See *IBM Operating System/2 Local Area Network Server Version 1.2: Network Administrator's Guide* for information about the format of separator files.

This feature is supported only on network servers. It can be set and queried on a local workstation, but no separator pages will be printed unless the server configuration of the LAN Manager program is installed on the computer.

prproc *(STRL)*
Default queue-processor.

parms *(STRL)*
Queue parameters.

This can be any text string or a NULL string.

comment *(STRL)*
Queue description.

A NULL string results in no comment. The maximum length is 48 characters.

status *(BIT16)*
Queue status.

PRQ3_PAUSED	Paused or not printing
PRQ3_PENDING	Active or deletion pending.

jobs *(COUNT2)*
Number of jobs in queue.

printerdestinations *(STRL)*
Print destinations connected to queue.

drivername *(STRL)*
Default device driver.

driverdata *(PDRIVDATA)*
Default driver data.

Note: An application can use **drivername**, **driverdata**, **prproc**, and **parms** to construct a valid DevOpenDC call based only on the queue name.

RENDERFILE

File-rendering structure.

dragfiles *(HWND)*
Conversation handle.

Created by DrgDragFiles.

source *(HSTR)*
Handle to source file name.

target *(HSTR)*
Handle to target file name.

move *(BOOL)*
Operation.

TRUE	Move the file.
FALSE	Copy the file.

reserved *(USHORT)*
Reserved.

SPLERR

Error value in the range 0 to 65 535.

For a list of possible return codes, see the chapter on the Spooler in the *Presentation Manager Programming Guide Technical Upgrade*.

Chapter 3. Device Function Calls

Summary of Changes

New	Updated	Section Title
	√	DevEscape
	√	DevOpenDC

DevEscape – Escape

On page 3-3, add the **DEVESC_SETMODE** escape to the **Code** parameter. Add the following description to the **Remarks** section:

DEVESC_SETMODE

Sets the printer into a particular mode. It is optional for printer drivers to support this escape, but those that do support it need to be aware of the code page of any built-in fonts. For example, if only code page 437 is built in, it is used if 437 is requested by DEVESC_SETMODE. However, if code page 865 is requested, a suitable code page/font could be downloaded.

This escape is metafiled and recorded.

InCount Number of bytes pointed to by **InData**
InData Buffer contains an *ESCSETMODE* structure (see page 8 in this book).
OutCount Not used; can be set to zero
OutData Not used; can be set to null.

DevOpenDC – Open Device Context

On page 3-10, replace the **Type** parameter definition with the following.

Type *(LONG)* – input
Type of device context:

OD_QUEUED A device, such as a printer or plotter, for which output is to be queued. Certain restrictions apply for this device type; see "Metafile Restrictions" in Appendix D.

OD_DIRECT A device, such as a printer or plotter, for which output is not to be queued.

OD_INFO A device, such as a printer or plotter, but the device context is used only to retrieve information (for example, font metrics). Drawing can be performed to a presentation space associated with such a device context, but no output medium is updated.

OD_METAFILE The device context is used to write a metafile. The presentation page defines the area of interest within the picture in the metafile. See OD_METAFILE_NOQUERY. Certain restrictions

© Copyright IBM Corp. 1990

	apply for this device type; see "Metafile Restrictions" in Appendix D.
OD_METAFILE_NOQUERY	The device context is used to write a metafile. Functionally, this device type is the same as OD_METAFILE, except that querying of attributes is not allowed with a presentation space while it is associated with an OD_METAFILE_NOQUERY device context. If querying of attributes is not required, OD_METAFILE_NOQUERY should be used in preference to OD_METAFILE, since it gives improved performance. Certain restrictions apply for this device type; see "Metafile Restrictions" in Appendix D.
OD_MEMORY	A device context that is used to contain a bit map. **Comp** identifies a device with which the memory device context is to be compatible.

Chapter 3.1. Drag Function Calls

Summary of Changes

New	Updated	Section Title
√		DrgAcceptDroppedFiles
√		DrgAccessDraginfo
√		DrgAddStrHandle
√		DrgAllocDraginfo
√		DrgAllocDragtransfer
√		DrgDeleteDraginfoStrHandles
√		DrgDeleteStrHandle
√		DrgDrag
√		DrgDragFiles
√		DrgFreeDraginfo
√		DrgFreeDragtransfer
√		DrgGetPS
√		DrgPostTransferMsg
√		DrgPushDraginfo
√		DrgQueryDragitem
√		DrgQueryDragitemCount
√		DrgQueryDragitemPtr
√		DrgQueryNativeRMF
√		DrgQueryNativeRMFLen
√		DrgQueryStrName
√		DrgQueryStrNameLen
√		DrgQueryTrueType
√		DrgQueryTrueTypeLen
√		DrgReleasePS
√		DrgSendTransferMsg
√		DrgSetDragImage
√		DrgSetDragitem
√		DrgSetDragPointer
√		DrgVerifyNativeRMF
√		DrgVerifyRMF
√		DrgVerifyTrueType
√		DrgVerifyType
√		DrgVerifyTypeSet

© Copyright IBM Corp. 1990

DrgAcceptDroppedFiles — Direct Manipulation for Files

This call handles the file direct manipulation protocol for a given window.

DrgAcceptDroppedFiles (Hwnd, Path, Types, DefaultOp, Reserved, *Success***)**

Parameters

Hwnd *(HWND)* — input
Handle of calling window.

Path *(STRL)* — input
Directory in which to place the dropped files.

If NULL, the files are placed in the current directory.

Types *(STRL)* — input
A list of types that are acceptable to the drop.

This string is of the form: TYPE[,TYPE...].

When this pointer is NULL, any type of file will be accepted.

DefaultOp *(USHORT)* — input
Default drag operation for this window.

The operation is either DO_MOVE or DO_DROP.

Reserved *(USHORT)* — input
Reserved.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion.
FALSE Error occurred.

Remarks

This call handles the file direct manipulation protocol for a given window. The window responds (DOR_DROP, **defaultop**) to DM_DRAGOVER messages for items with a type matching the acceptable type string and with a rendering mechanism and format of <DRM_OS2FILE,DRF_UNKNOWN>. Not all dragged objects need match this criteria for the drop to be acceptable.

After the drop occurs, this call handles the conversation required to complete the direct manipulation operation for all acceptable objects. A DM_ENDCONVERSATION (DMFL_TARGETFAIL) message is sent to the source when an object is unacceptable.

When an error occurs during a move or copy, the caller is sent a DM_DRAGERROR message. The caller can take corrective action.

As the move or copy operation is successfully completed for each file, a DM_DRAGFILECOMPLETE message is sent to the caller. No message is sent when the operation fails.

The call returns TRUE if the operation is successful and FALSE if an error occurs.

This call requires the existence of a message queue.

DrgAccessDraginfo – Access Drag Information

This call accesses a *DRAGINFO* structure.

DrgAccessDraginfo (Draginfo, *Success***)**

Parameters

Draginfo *(DRAGINFO)* – input
Drag-information structure.

Success *(BOOL)* – return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_ACCESS_DENIED

Remarks

This call is used by the target of a drag operation to access a *DRAGINFO* structure. The address of the structure is passed in a drag message (DM_DRAGOVER, DM_DROP, or DM_DROPHELP).

To release the structure, use the DrgFreeDraginfo call.

This call requires the existence of a message queue.

DrgAddStrHandle — Create String Handle

This call creates a handle to a string.

DrgAddStrHandle (String, *Hstr*)

Parameters

String *(STRL)* — input
String for which a handle is to be created.

Hstr *(HSTR)* — return
Handle of the string:

NULL Error occurred
Other String handle created.

Possible returns from WinGetLastError:

PMERR_INVALID_PARAMETER
PMERR_RESOURCE_DEPLETION

Remarks

The handle can be used by any application to reference the input string.

This call must be made by the source of a drag whenever a string is to be passed in a *DRAGINFO* structure.

This call requires the existence of a message queue.

DrgAllocDraginfo — Allocate DRAGINFO Structure

This call allocates a *DRAGINFO* structure.

DrgAllocDraginfo (Ditem, *Draginfo***)**

Parameters

Ditem *(COUNT2)* — input
Number of objects being dragged.

This number must be greater than zero.

Draginfo *(DRAGINFO)* — return
DRAGINFO structure.

NULL Error occurred
Other The *DRAGINFO* structure.

Possible returns from WinGetLastError:

PMERR_INSUFFICIENT_MEMORY
PMERR_INVALID_PARAMETER

Remarks

This call must be made prior to calling DrgDrag.

The caller should not alter any part of the *DRAGINFO* structure. The *DRAGITEM* structures associated with a *DRAGINFO* structure should be altered only with DrgSetDragitem or with a pointer obtained with DrgQueryDragitemPtr.

This call requires the existence of a message queue.

DrgAllocDragtransfer – Allocate DRAGTRANSFER Structures

This call allocates a specified number of *DRAGTRANSFER* structures from a single segment.

DrgAllocDragtransfer (Dxfer, *Result*)

Parameters

Dxfer *(COUNT2)* – input
The number of *DRAGTRANSFER* structures to be allocated.

This number must be greater than zero.

Result *(DRAGTRANSFER)* – return
Pointer to an array of *DRAGTRANSFER* structures.

NULL Error occurred
Other The array of *DRAGTRANSFER* structures.

Possible returns from WinGetLastError:

PMERR_MEMORY_ALLOCATION_ERR
PMERR_INSUFFICIENT_MEMORY
PMERR_PARAMETER_OUT_OF_RANGE

Remarks

This call must be made prior to sending a DM_RENDER message.

This call requires the existence of a message queue.

DrgDeleteDraginfoStrHandles — Delete DRAGINFO String Handles

This call deletes each unique string handle in a *DRAGINFO* structure.

DrgDeleteDraginfoStrHandles (DragInfo, *Success*)

Parameters

DragInfo *(DRAGINFO)* — input
The *DRAGINFO* structure containing string handles to delete.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_INVALID_PARAMETER

Remarks

Using this call is equivalent to calling DrgDeleteStrHandle for each unique string in a *DRAGINFO* structure.

This call must be made by the target of a direct manipulation operation either:

- After processing a DM_DROPHELP message
 or
- After completing the direct manipulation operation begun as a result of a DM_DROP message.

This call requires the existence of a message queue.

DrgDeleteStrHandle — Delete String Handle

This call deletes a string handle.

DrgDeleteStrHandle (Hstr, *Success*)

Parameters

Hstr *(HSTR)* — input
The string handle to delete.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_INVALID_PARAMETER

Remarks

This call must be used to delete a string handle created by DrgAddStrHandle.

This call requires the existence of a message queue.

DrgDrag — Drag

This call performs a drag operation.

DrgDrag (Source, Draginfo, pdimg, cdimg, VkTerminate, Reserved, *Dest*)

Parameters

Source *(HWND)* — input
Handle of the window calling DrgDrag.

This window is the source of the drag.

Draginfo *(DRAGINFO)* — input/output
A *DRAGINFO* structure.

pdimg *(DRAGIMAGE)* — input
Pointer to an array of *DRAGIMAGE* structures.

These structures describe the images that are to be drawn under the mouse pointer during the drag.

cdimg *(USHORT)* — input
Size of the **pdimg** array.

VkTerminate *(SHORT)* — input
Mouse button that terminates the drag operation.

VK_BUTTON1 Release of button 1 terminates the drag.
VK_BUTTON2 Release of button 2 terminates the drag.
VK_BUTTON3 Release of button 3 terminates the drag.

Reserved *(STORAGE)* — input
Reserved.

Must be set to NULL by the caller.

Dest *(HWND)* — return
Handle of window the dragged objects were dropped on.

NULL Error occurred
Other Window handle.

Possible returns from WinGetLastError:

PMERR_INVALID_HWND
PMERR_INVALID_PARAMETER
PMERR_INSUFFICIENT_MEMORY

Remarks

This call:

- Initiates a direct manipulation operation
- Uses the input image to provide visual feedback to the user
- Notifies other windows as the dragged object passes over
- Notifies the destination if the object is dropped.

DrgDrag is called when the system direct manipulation button is pressed while over a window and a mouse movement follows. As the mouse pointer moves over a potential target, a DM_DRAGOVER message is sent to the target. When the mouse pointer moves from one target window to another, a DM_DRAGLEAVE message is sent to the former target and the target highlighting is removed.

If the mouse pointer is over a valid target when the drag button is released, a DM_DROP message is sent to the target.

Before the DM_DROP message is sent, the **xoffset** and **yoffset** fields are copied from the *DRAGIMAGE* structures to the corresponding fields in the *DRAGITEM* structures. The values from the first *DRAGIMAGE* are copied to the first *DRAGITEM*, from the second *DRAGIMAGE* to the second *DRAGITEM*, and so on. The target can use this information to place the images in the same spatial relationship after the drop. If there are more *DRAGITEM* structures than there are *DRAGIMAGE* structures, the **xoffset** and **yoffset** from the final *DRAGIMAGE* are placed in each of the remaining *DRAGITEM* structures.

The following keys are active during the drag operation:

Esc The drag operation is canceled.
F1 A DM_DROPHELP message is posted to the target so that it can provide context help for the drag operation. The drag operation is canceled.

Before invoking DrgDrag, the caller is responsible for:

- Obtaining a *DRAGINFO* structure using DrgAllocDraginfo
- Initializing the *DRAGINFO* structure using DrgSetDragitem.

On return from DrgDrag, the caller must free the structure using DrgFreeDraginfo.

If the dragged objects are not dropped, NULL is returned.

This call requires the existence of a message queue.

DrgDragFiles — Begin Dragging Files

This call begins a direct manipulation operation for one or more files.

DrgDragFiles (Hwnd, Files, Types, Targets, cFiles, Drag, Term, SourceRender, Reserved, *Success***)**

Parameters

Hwnd *(HWND)* — input
Handle of calling window.

Files *(PAPSZ)* — input
The names of the files to be dragged.

Types *(PAPSZ)* — input
The file types of the files to be dragged.

Targets *(PAPSZ)* — input
The target file names.

cFiles *(COUNT2)* — input
Number of files to be dragged.

Drag *(HPOINTER)* — input
Icon to display during the drag.

Term *(USHORT)* — input
Button that terminates the drag.

SourceRender *(BOOL)* — input
Flag to indicate whether the source must perform the move or copy.

TRUE The caller will receive a DM_RENDERFILE message for each file.
FALSE All file manipulation is performed by DrgDragFiles.

Reserved *(ULONG)* — input
Reserved.

Success *(BOOL)* — return
Success indicator:

TRUE The drag operation was initiated successfully.
FALSE An error occurred.

Remarks

This call begins a direct manipulation operation for one or more files. *DRAGINFO* and *DRAGITEM* structures are allocated and initialized, and are then used as input to DrgDrag. All of the post-drag conversation required to complete the direct manipulation operation is handled by an object window created by this call.

The caller should set **SourceRender** to TRUE if it must perform the file manipulation for any of these files. When **SourceRender** is TRUE, the caller receives a DM_RENDERFILE message as the drag-object window receives a DM_RENDER message. The caller should move or copy the file after receiving the DM_RENDERFILE message. The caller should then send a DM_FILERENDERED

message to the drag-object window, and the drag-object window should send a DM_RENDERCOMPLETE message to the target.

When **Types** is NULL, the .TYPE EA is interrogated to determine the type for each file in **Files**. When **Types** is not NULL, the size of the array is expected to be the same as the size of **Files**. When any individual pointer in the array is NULL, the .TYPE EA for the corresponding file is read. When .TYPE EA does not exist for any file for which it is needed, a type of DRT_UNKNOWN is used.

When **Targets** is NULL, the target name for a file will be the same as the source file name with the path information removed. If **Targets** is not NULL, the size of the array is expected to be the same as the size of **Files**. If any individual pointer in the array is NULL, the target name for the corresponding file will match the source name minus the path information.

The rendering mechanism and format for each file is:
<DRM_OS2FILE,DRF_UNKNOWN>.

When an error occurs during the move or copy, the caller is sent a DM_DRAGERROR message. The caller can take corrective action.

As the operation is complete for each file in the list, a DM_DRAGFILECOMPLETE message is sent to the caller of DrgDragFiles. The caller is thus notified that resources can be freed for a particular file.

This call returns TRUE if the drag operation was initiated successfully and FALSE if an error occurred.

This call requires the existence of a message queue.

DrgFreeDraginfo — Free DRAGINFO structure

This call frees a *DRAGINFO* structure allocated by DrgAllocDraginfo.

DrgFreeDraginfo (DragInfo, *Success***)**

Parameters

DragInfo *(DRAGINFO)* — input
A *DRAGINFO* structure.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_MEMORY_DEALLOCATION_ERR
PMERR_SOURCE_SAME_AS_TARGET

Remarks

DrgFreeDraginfo fails with an error of PMERR_SOURCE_SAME_AS_TARGET if it is called by the process that called DrgDrag before DrgDrag returns. When a process is performing a drag operation between two of its own windows, this prevents the source window from freeing the *DRAGINFO* structure before the target window finishes processing.

This call requires the existence of a message queue.

DrgFreeDragtransfer — Free DRAGTRANSFER Storage

This call frees the storage associated with a *DRAGTRANSFER* structure.

DrgFreeDragtransfer (dxfer, *Rc*)

Parameters

dxfer *(DRAGTRANSFER)* — input
The *DRAGTRANSFER* structures to be freed.

Rc *(BOOL)* — return
Return code:

0 The structure was freed.
Other Deallocation failed.

Possible returns from WinGetLastError:

PMERR_MEMORY_DEALLOCATION_ERR

Remarks

This call frees the *DRAGTRANSFER* structures allocated by calls to DrgAllocDragtransfer. When all of the *DRAGTRANSFER* structures have been freed, the memory block containing the *DRAGTRANSFER* array is deallocated.

This call requires the existence of a message queue.

DrgGetPS — Get Drag Presentation Space

This call gets a presentation space that is used to provide target feedback to the user during a drag operation.

DrgGetPS (Hwnd, *Hps*)

Parameters

Hwnd *(HWND)* — input
Handle of the window for which presentation space is required.

Hps *(HPS)* — return
Presentation-space handle used for drawing in the window.

NULL Error occurred.

Possible returns from WinGetLastError:

PMERR_INVALID_HWND
PMERR_NOT_DRAGGING

Remarks

This call returns a handle to a presentation space that can be used for drawing while a direct manipulation operation is in progress.

DrgGetPS is called only during a direct manipulation operation. This call is made only after a DM_DRAGOVER, DM_DRAGLEAVE, or DM_DROP message has been received.

In order to draw target emphasis, an application must use DrgGetPS and DrgReleasePS to unlock its window.

The presentation space created with DrgGetPS must be freed with DrgReleasePS.

This call requires the existence of a message queue.

DrgPostTransferMsg — Post Drag Message

This call posts a message to the other application involved in the direct manipulation operation.

DrgPostTransferMsg (To, Msgid, dxfer, Fs, Reserved, Retry, *Success***)**

Parameters

To *(HWND)* — input
Window handle to which the message is to be posted:

Target **Item** in the *DRAGITEM* structure
Source **cIient** in the *DRAGTRANSFER* structure.

Msgid *(USHORT)* — input
Identifier of the message to be posted.

DM_RENDERCOMPLETE is the only valid message.

dxfer *(DRAGTRANSFER)* — input
The *DRAGTRANSFER* structure.

Fs *(USHORT)* — input
The flags to be passed in the **param2** parameter of the message.

Reserved *(USHORT)* — input
Reserved.

This must be 0.

Retry *(BOOL)* — input
Retry indicator:

TRUE If the destination queue is full, the message posting is retried at 1-second intervals until the message is posted successfully.

In this case, DrgPostTransferMsg dispatches any messages in the queue by calling WinPeekMsg and WinDispatchMsg in a loop. The application can receive messages sent by other applications while it is trying to post drag transfer messages.

FALSE The call returns FALSE without retrying.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Remarks

The **reply** field in the *DRAGTRANSFER* structure is set to 0 before the message is posted. If the posting fails for any reason, FALSE is returned.

This call requires the existence of a message queue.

DrgPushDraginfo — Access a DRAGINFO Structure

This call gives a process access to a *DRAGINFO* structure.

DrgPushDraginfo (Draginfo, Dest, *Success***)**

Parameters

Draginfo *(DRAGINFO)* — input
DRAGINFO structure.

Dest *(HWND)* — input
Handle of the window whose process is to be given access to a *DRAGINFO* structure.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_ACCESS_DENIED
PMERR_INSUFFICIENT_MEMORY

Remarks

The receiving process is responsible for:

1. Deleting the string handles in the *DRAGINFO* structure with DrgDeleteDraginfoStrHandles
2. Freeing the *DRAGINFO* structure using DrgFreeDraginfo.

This call requires the existence of a message queue.

DrgQueryDragitem — Get DRAGITEM Structure

This call returns a *DRAGITEM* structure used in the direct manipulation operation.

DrgQueryDragitem (DragInfo, Buffer, DragItem, Item, *Success***)**

Parameters

DragInfo *(DRAGINFO)* — input
The *DRAGINFO* structure from which the *DRAGITEM* structure is obtained.

Buffer *(COUNT2B)* — input
Maximum number of bytes to copy.

DragItem *(DRAGITEM)* — output
Buffer into which the *DRAGITEM* structure is copied.

Item *(COUNT2)* — input
Zero-based index of the *DRAGITEM* to be returned.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Remarks

This call returns the *DRAGITEM* structure identified by **Item**.

DrgQueryDragitemCount — Get Dragged Object Count

This call returns the number of objects being dragged during the current direct manipulation operation.

DrgQueryDragItemCount (DragInfo, *Ditem***)**

Parameters

DragInfo *(DRAGINFO)* — input
The *DRAGINFO* structure for which the number of dragged objects is requested.

Ditem *(COUNT2)* — return
Number of objects being dragged.

DrgQueryDragitemPtr — Get DRAGITEM Structure

This call returns a pointer to the *DRAGITEM* structure used in the direct manipulation operation.

DrgQueryDragitemPtr (DragInfo, n, *Dragitem***)**

Parameters

DragInfo *(DRAGINFO)* — input
The *DRAGINFO* structure from which the *DRAGITEM* structure is obtained.

n *(USHORT)* — input
Zero-based index of the *DRAGITEM* structure for which the pointer is to be returned.

DragItem *(DRAGITEM)* — return
The *DRAGITEM* structure.

Remarks

This call returns a pointer to **itemId** in the *DRAGITEM* structure used in the direct manipulation operation.

DrgQueryNativeRMF — Get Format of a Dragged Object

This call obtains the ordered pair that represents the native rendering mechanism and format of the dragged object.

DrgQueryNativeRMF (DragItem, BufLen, Buffer, *Success***)**

Parameters

DragItem *(DRAGITEM)* — input
DRAGITEM structure whose native rendering mechanism and format is to be obtained.

BufLen *(COUNT2B)* — input
Maximum number of bytes to copy to the buffer.

Buffer *(CHAR)* — output
Buffer in which the null-terminated string is to be returned.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_INVALID_PARAMETER

Remarks

If the rendering mechanism and format string for the object are NULL, FALSE is returned. If TRUE is returned, the format of the string is: <MECHANISM,FORMAT>.

The native rendering mechanism and format are the first ordered pair, or the first ordered pair produced by a cross product, in the string associated with **rmf** in the *DRAGITEM* structure.

DrgQueryNativeRMFLen can be used to determine the size of the buffer required to hold the string returned by this call.

This call requires the existence of a message queue.

DrgQueryNativeRMFLen — Get String Length of a Dragged Object

This call obtains the length of the string representing the rendering mechanism and format of the dragged object.

DrgQueryNativeRMFLen (DragItem, *Length***)**

Parameters

DragItem *(DRAGITEM)* — input
DRAGITEM structure whose native rendering mechanism and format string length are to be obtained.

Length *(USHORT)* — return
String length of the ordered pair:

0 Error occurred
Other String length of the ordered pair, excluding the null-terminating byte.

Possible returns from WinGetLastError:

PMERR_INVALID_PARAMETER

Remarks

This call is used to determine the size of the buffer that contains the string representing the native rendering mechanism and format of the dragged object.

If the input string handle is NULL or not valid, a length of 0 is returned.

This call requires the existence of a message queue.

DrgQueryStrName – Get String Contents

This call gets the contents of a string associated with a string handle.

DrgQueryStrName (Hstr, Buflen, Buffer, *Length***)**

Parameters

Hstr *(HSTR)* – input
Handle to a string.

The handle must have been created with DrgAddStrHandle.

Buflen *(COUNT2B)* – input
Maximum number of bytes to copy.

Buffer *(STRL)* – output
Buffer where the null-terminated string is returned.

Length *(USHORT)* – return
Number of bytes written to **Buffer**.

Possible returns from WinGetLastError:

PMERR_INVALID_PARAMETER

Remarks

This call should be made whenever the contents of a string referenced by a drag string handle are required. If the input string handle is NULL or not valid, a null string is returned.

This call requires the existence of a message queue.

DrgQueryStrNameLen – Get String Length

This call gets the length of a string associated with a string handle.

DrgQueryStrNameLen (Hstr, *Length*)

Parameters

Hstr *(HSTR)* – input
String handle.

The handle must be created with DrgAddStrHandle.

Length *(USHORT)* – return
String length.

0 The string handle is NULL or is not valid.
Other The length of the string associated with the string handle, excluding the null terminating byte.

Possible returns from WinGetLastError:

PMERR_INVALID_PARAMETER

Remarks

This call should be made prior to DrgQueryStrName. It is used to determine and allocate the buffer size for the string associated with the string handle. If the input string handle is NULL or not valid, a length of 0 is returned.

This call requires the existence of a message queue.

DrgQueryTrueType — Get True Type of Dragged Object

This call obtains the true type of a dragged object.

DrgQueryTrueType (DragItem, BufLen, Buffer, *Success***)**

Parameters

DragItem *(DRAGITEM)* — input
DRAGITEM structure whose true type is to be obtained.

BufLen *(COUNT2B)* — input
Maximum number of bytes to copy to the buffer.

Buffer *(STRL)* — output
Buffer in which the null-terminated string is to be returned.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_INVALID_PARAMETER

Remarks

The true type of an object is the first type in the string referenced by **type** in the *DRAGITEM* structure.

This call can be made after calling DrgQueryTrueTypeLen. If the type string for the object is NULL, FALSE is returned.

This call requires the existence of a message queue.

DrgQueryTrueTypeLen — Get String Length of Dragged Object

This call obtains the length of the string that represents the true type of a dragged object.

DrgQueryTrueTypeLen (DragItem, *Length***)**

Parameters

DragItem *(DRAGITEM)* — input
DRAGITEM structure whose true type length is to be obtained.

Length *(USHORT)* — return
String length:

0 Error occurred
Other The length of the first element of the character string associated with **type**, excluding the null-terminating byte.

Possible returns from WinGetLastError:

PMERR_INVALID_PARAMETER

Remarks

This call can be used to determine the buffer size to allocate for the string representing the true type of a dragged object. The true type of an object is the first type in the type string referenced by **type** in the *DRAGITEM* structure.

This call can be made prior to DrgQueryTrueType.

If the input string handle is NULL or not valid, a length of 0 is returned.

This call requires the existence of a message queue.

DrgReleasePS — Release Presentation Space

This call releases a presentation space obtained by using DrgGetPS.

DrgReleasePS (Hps, *Success*)

Parameters

Hps *(HPS)* — input
Handle of the presentation space to release.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_INV_HPS
PMERR_NOT_DRAGGING

Remarks

Only presentation spaces created with DrgGetPS can be released using this call.

The presentation-space handle should not be used after this call.

This call requires the existence of a message queue.

DrgSendTransferMsg — Send Drag Message

This call sends a message to the other application involved in the direct manipulation operation.

DrgSendTransferMsg (To, Msgid, Param1, Param2, *Reply***)**

Parameters

To *(HWND)* — input
Window handle to which the message is to be sent:

Target **Item** in the *DRAGITEM* structure
Source **cItem** in the *DRAGTRANSFER* structure.

Msgid *(USHORT)* — input
Identifier of the message to be sent.

Valid messages are:

DM_ENDCONVERSATION
DM_PRINT
DM_RENDER
DM_RENDERPREPARE

Param1 *(MPARAM)* — input
mp1 for the message.

Param2 *(MPARAM)* — input
mp2 for the message.

Reply *(MRESULT)* — return
Message-return data.

Remarks

If the message to be sent is DM_RENDER or DM_RENDERCOMPLETE, the **reply** field in *DRAGTRANSFER* is set to 0 before the message is sent. If the message cannot be sent, FALSE is returned.

When the message to be sent is DM_RENDER, DosGiveSeg is called. DosGiveSeg gives access to the *DRAGTRANSFER* structure to the process that owns the window indicated by **To**. The use count for the segment in which the *DRAGTRANSFER* structure exists is incremented.

The process to which the message is being sent must call DrgFreeDragtransfer for the *DRAGTRANSFER* structure before the segment can be freed.

This call requires the existence of a message queue.

DrgSetDragImage — Set Drag Image

This call sets the image that is being dragged.

DrgSetDragImage (DragInfo, pdimg, cdimg, Reserved, *Success***)**

Parameters

DragInfo *(DRAGINFO)* — input
DRAGINFO structure representing the drag operation for which the mouse pointer is to be set.

pdimg *(DRAGIMAGE)* — input
Pointer to an array of *DRAGIMAGE* structures.

These structures describe the images to be drawn under the mouse pointer during the drag.

cdimg *(USHORT)* — input
Size of the **pdimg** array.

Reserved *(STORAGE)* — input
Reserved.

Must be set to NULL by the caller.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_ACCESS_DENIED
PMERR_INVALID_PARAMETER
PMERR_INSUFFICIENT_MEMORY

Remarks

The image that is set with DrgSetDragImage is used only while the pointer is over the target that made the call. If the pointer leaves the original target, the new target can specify an image by calling DrgSetDragImage.

If the new target does not call DrgSetDragImage, the original image that was supplied on the call to DrgDrag is used.

This call requires the existence of a message queue.

DrgSetDragitem — Set Values in DRAGITEM

This call sets the values in a *DRAGITEM* structure.

DrgSetDragItem (DragInfo, DragItem, Buffer, lItem, *Success***)**

Parameters

DragInfo *(DRAGINFO)* — input
DRAGINFO structure in which to place the *DRAGITEM*.

DragItem *(DRAGITEM)* — input
DRAGITEM structure to place in *DRAGINFO*.

Buffer *(COUNT2B)* — input
Size of the *DRAGITEM* addressed by **DragItem**.

lItem *(USHORT)* — input
Zero-based index of the *DRAGITEM* to be set.

Success *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Remarks

This call is used to initialize the *DRAGINFO* structure before calling DrgDrag.

This call is used only by the source of the drag, not by the target.

DrgSetDragPointer — Set Mouse Pointer

This call sets the pointer to be used while over the current target.

DrgSetDragPointer (DragInfo, Handle, *Result*)

Parameters

DragInfo *(DRAGINFO)* — input
Pointer to the *DRAGINFO* structure to be used for this drag.

Handle *(HPOINTER)* — input
Handle to the pointer to use.

Result *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_INVALID_HPTR

Remarks

This call sets the pointer to be used to indicate the hot spot while dragging over the current target.

The pointer that is set with DrgSetDragPointer is used only while it is over the current target. The pointer is reset to the default when a new target is dragged over.

This call can be used by an application to provide meaningful augmentation emphasis for an operation that is unknown to the system (for example, swap).

When the drag pointer is successfully set, TRUE is returned.

This call requires the existence of a message queue.

DrgVerifyNativeRMF — Verify Rendering Mechanism and Format

This call determines if the native rendering mechanism and format of an object match any supplied by the application.

DrgVerifyNativeRMF (DragItem, RMF, *Valid*)

Parameters

DragItem *(DRAGITEM)* — input
DRAGITEM structure whose native rendering mechanism and format are to be verified.

RMF *(STRL)* — input
A string specifying the rendering mechanism and format.

The string is of the form: MECHFMT[,MECHFMT,MECHFMT,...], where MECHFMT can be in either of these formats:

- <mechanism(1),format(1)>
- (mechanism(1)[, mechanism(n)...]) (format(1)[,format(n)...])

Valid *(BOOL)* — return
Validity indicator:

TRUE Successful completion
FALSE Error occurred.

Remarks

This call determines if the native rendering mechanism and format of a dragged object are understood by the target.

If TRUE is returned, the target may be able to carry out the action indicated by the direct manipulation itself, or it can select the native rendering mechanism and format as those to be used for the data exchange.

This call requires the existence of a message queue.

DrgVerifyRMF — Verify Rendering Mechanism and Format

This call determines if a given rendering mechanism and format are supported for a dragged object.

DrgVerifyRMF (DragItem, Mech, Format, *Valid***)**

Parameters

DragItem *(DRAGITEM)* — input
DRAGITEM structure whose native rendering mechanism and format is to be validated.

Mech *(STRL)* — input
A string specifying the rendering mechanism to search for.

Format *(STRL)* — input
A string specifying the rendering format to search for.

Valid *(BOOL)* — return
Validity indicator:

TRUE Successful completion
FALSE Error occurred.

Remarks

This call determines if a given rendering mechanism and format ordered pair are represented in the set of valid pairs specified by **rmf** for the dragged object.

This call requires the existence of a message queue.

DrgVerifyTrueType — Verify True Type of Dragged Object

This call determines if the true type of a dragged object matches an application-supplied type string.

DrgVerifyTrueType (DragItem, Type, *Valid*)

Parameters

DragItem *(DRAGITEM)* — input
DRAGITEM structure whose true type is to be verified.

Type *(STRL)* — output
A string specifying a type.

This string is in the format: TYPE[,TYPE...].

Valid *(BOOL)* — return
Validity indicator:

TRUE Successful completion
FALSE Error occurred.

Remarks

If an item in the string pointed to by **Type** matches the first type in the string associated with **type** in the *DRAGITEM* structure, TRUE is returned.

A target application uses this call to determine if it supports the true type of a dragged object. If the application does not support the true type, it can either disallow a drop or change its default operation. If the default operation is a move, the drop should be disallowed, or the operation changed to a copy to prevent any loss of data for the object.

This call requires the existence of a message queue.

DrgVerifyType — Verify Type of Dragged Object

This call verifies whether a given type is present in the list of types defined for a drag object.

DrgVerifyType (DragItem, Type, *Valid***)**

Parameters

DragItem *(DRAGITEM)* — input
DRAGITEM structure whose **type** is to be verified.

Type *(STRL)* — input
A string specifying the types to search for.

This string is in the format: TYPE[,TYPE...].

Valid *(BOOL)* — return
Success indicator:

TRUE Successful completion
FALSE Error occurred.

Possible returns from WinGetLastError:

PMERR_INVALID_PARAMETER
PMERR_INSUFFICIENT_MEMORY

Remarks

If at least one of the types specified by **Type** is present in **type** in the *DRAGITEM* structure, TRUE is returned.

This call requires the existence of a message queue.

DrgVerifyTypeSet — Verify Types

This call returns the intersection of the contents of the string associated with the type-string handle for an object and an application-specified type string.

DrgVerifyTypeSet (DragItem, Type, BufLen, Buffer, *Match***)**

Parameters

DragItem *(DRAGITEM)* — input
DRAGITEM structure whose **type** is to be verified.

Type *(STRL)* — input
A string specifying the types to search for.

This string is in the format: TYPE[,TYPE...].

BufLen *(COUNT2B)* — input
Size of the return buffer.

The buffer should be at least one byte longer than the length of the string pointed to by **Type**.

Buffer *(STRL)* — output
Buffer where the intersection string is returned.

Match *(BOOL)* — return
Match indicator:

TRUE Successful completion
FALSE Error occurred.

Remarks

If at least one of the types specified by **Type** is present in **type** in the *DRAGITEM* structure, TRUE is returned.

This call requires the existence of a message queue.

Chapter 4. Graphics Function Calls

Summary of Changes

New	Updated	Section Title
	√	GpiAssociate
	√	GpiCharStringPos
	√	GpiCharStringPosAt
	√	GpiCreateLogColorTable
	√	GpiCreateLogFont
	√	GpiDeleteBitmap
	√	GpiPlayMetafile
	√	GpiQueryBitmapParameters
	√	GpiQueryCharStringPos
	√	GpiQueryCharStringPosAt
	√	GpiQueryDefCharBox
	√	GpiQueryLogColorTable
	√	GpiQueryTextBox
	√	GpiRealizeColorTable
	√	GpiResetPS
	√	GpiSetClipRegion
	√	GpiSetMarkerSet
	√	GpiSetMetaFileBits
	√	GpiSetPageViewport
	√	GpiSetPatternSet
	√	GpiUnrealizeColorTable

GpiAssociate — Associate

On page 4-10, add the following paragraph to the Remarks section:

GpiUnrealizeColorTable should be issued before a presentation space that has a realized logical color table is disassociated or reassociated, that is before any GpiAssociate call referencing that presentation space.

GpiCharStringPos — Character String Position

On page 4-30, add the following to the **Adx** parameter:

Any negative values are treated as if they were zero.

© Copyright IBM Corp. 1990

GpiCharStringPosAt — Character String Position At

On page 4-32, change the description of the **CHS_LEAVEPOS** option of the **Options** parameter to:

CHS_LEAVEPOS If set, current position is unchanged by this function. If not set, current position is moved to the position at which the next character would have been drawn, had there been one.

Also, add the following to the **Adx** parameter:

Any negative values are treated as if they were zero.

GpiCreateLogColorTable — Create Logical Color Table

On page 4-50, add the following to LCOLF_INDRGB in the **Format** parameter:

Each index specified must be greater than or equal to zero.

Add the following to the **Start** parameter:

The starting index must be greater than or equal to zero.

On page 4-52, replace the paragraph of the Remarks section beginning 'If LCOL_REALIZABLE is set, but the currently associated device does not support realization...' with:

If LCOL_REALIZABLE is set, but the currently associated device does not support realization, a warning is raised (not an error; the value TRUE is returned). The LCOL_REALIZABLE property of a logical color table is lost if the presentation space is reassociated. It is necessary to create the logical color table again in order to reestablish the realizable property. This is only necessary if the presentation space is associated with a device that supports realization.

If a color table with entries at zero and the highest physical index allowed on the device is realized, these two entries must have different RGB values, to ensure that the pointer is visible against any color.

GpiCreateLogFont — Create Logical Font

On page 4-55, in the tenth paragraph of the Remarks section, note that the values of the character box height and width for a font are held in the **emheight** and **eminc** fields of the *FONTMETRICS* structure (not **maxbaselineext** and **avecharwidth**).

GpiDeleteBitmap — Delete Bit Map

On page 4-60, add the following:

Remarks

There are restrictions on the use of this function while generating a metafile or a PM_Q_STD print file; see "Metafile Restrictions" in Appendix D.

GpiPlayMetafile — Play Metafile

On page 4-112, in the **Optarray (PMF_COLORREALIZABLE)** parameter description, replace **CREA_REALIZE** with:

CREA_DOREALIZE
Load the color table with the realizable option set, and realize the color table.

GpiQueryBitmapParameters — Query Bit-Map Parameters

On page 4-138, change the **Data** parameter to input/output.

Add this programming note:

The **length** field of the *BITMAPINFOHEADER* structure *can* be set by the application before making this call, but this is not mandatory.

This note replaces the one described in the READ.ME file in the OS/2 Version 1.2 Programming Tools.

GpiQueryCharStringPos — Query Character String Positions

On page 4-146, add the following to the **Xincrements** parameter:

Any negative values are treated as if they were zero.

GpiQueryCharStringPosAt — Query Character String Positions At

On page 4-147, add the following to the **Xincrements** parameter:

Any negative values are treated as if they were zero.

GpiQueryDefCharBox — Query Default Graphics Character Box

On page 4-159, add the following:

Remarks
The values returned are the same as the initial default value of the character-box attribute. See GpiSetCharBox for further information.

GpiQueryLogColorTable — Query Logical Color Table

On page 4-182, add the following to the **Start** parameter:

The starting index must be greater than or equal to zero.

Also, change CLR_NOINDEX in the description of the **Array** parameter to QLCT_NOTLOADED.

GpiQueryTextBox — Query Text Box

On page 4-213, add the following to the Programming Note in the Remarks section:

The dimensions of the box do not correspond directly to those of the character box (see GpiSetCharBox).

GpiRealizeColorTable — Realize Color Table

On page 4-217, change the second paragraph of the Remarks section to:

If the presentation space is currently associated with a screen window device, this call should only be issued when the associated window is maximized and has the focus.

Also add the following paragraph:

After calling GpiRealizeColorTable, the application should repaint the screen. Note that this does not change the colors of icons or other bit maps.

GpiResetPS — Reset Presentation Space

On page 4-223, add the following item to the list in the **GRES_ALL** option of the **Options** parameter:

- The pick aperture size and position are reset to the default.

On page 4-224, add the following paragraph to the Remarks section:

This function has no effect on the realization state of the color table. If a color table is realized, the application should issue GpiUnrealizeColorTable before GpiResetPS.

GpiSetCharBox — Set Character Box

On page 4-254, replace the forth paragraph of the Remarks section with:

For outline fonts, characters are defined in font definition space. The **xdeviceres** and **ydeviceres** fields (see *FONTMETRICS*) define a rectangle in font definition space that is mapped to the character box rectangle (modified by the character angle and shear attributes) in world coordinates. **ydeviceres** corresponds to the font point size in font definition space, and therefore the character box height corresponds to the font point size in world coordinates. This is typically less than the **maxbaselineext**.

The effective width of each character from an outline font is the character box width, scaled by the ratio of that character's width to **xdeviceres**. The **avecharwidth** (for a proportionally-spaced font) will therefore typically be smaller than the character box width. Indeed, because of differences in font design, **avecharwidth** and **maxbaselineext** vary between different fonts, even when the character box dimensions are the same.

emInc and **emHeight** are always equal to the character box width and height, respectively.

GpiSetClipRegion — Set Clip Region

On page 4-266, add the following to the Remarks section:

It is the responsibility of the application to free the previous clip region (if any) with GpiDestroyRegion, even if this region was not originally created explicitly by the application.

GpiSetMarkerSet — Set Marker Set

On page 4-302, add the following to the Remarks section:

If the default marker set is changed (using GpiSetDefAttrs), the initial default marker set cannot be selected with GpiSetMarkerSet.

GpiSetMetaFileBits — Set Metafile Bits

On page 4-303, add the following to the Remarks section:

The length of the metafile is increased, if necessary, to accommodate the supplied data. If the supplied data is shorter, the metafile length is not reduced. However, in this case the metafile is still valid, providing the data in it is complete and otherwise correct.

GpiSetPageViewport — Set Page Viewport

On page 4-308, add the following paragraph to the Remarks section:

This call is ignored if issued to a presentation space that is associated with a device context of type OD_QUEUED (with PM_Q_STD data), OD_METAFILE, or OD_METAFILE_NOQUERY.

GpiSetPatternSet — Set Pattern Set

On page 4-313, add the following to the Remarks section:

If the default pattern set is changed (using GpiSetDefAttrs), the initial default marker set cannot be selected with GpiSetPatternSet.

GpiUnrealizeColorTable — Unrealize Color Table

On page 4-338, change the Remarks section to:

This call has the opposite effect of GpiRealizeColorTable. It causes the default device physical color table to be reinstated, and should be issued when the associated window ceases to be maximized or loses the focus.

The logical color table remains unchanged.

After making this call, the application must ensure that the screen is repainted.

Chapter 5. Picture Function Calls

Summary of Changes

New	Updated	Section Title
	√	PicIchg
	√	PicPrint

PicIchg – Picture Interchange Convert

On page 5-2, replace the text for item 1 of the Remarks section with:

EBCDIC Code page.

Text strings and symbol set names in PIF files are expressed in EBCDIC. Also, symbol sets define codepoints according to an EBCDIC codepage. PicIchg therefore needs to know the EBCDIC codepage number to use for the conversions. It derives this from the country of origin of the PIF or symbol set.

By default the country of origin is assumed to be the country returned by the DosGetCtryInfo call. However, it can be overridden by running the Convert Picture File utility (PICICHG.EXE), and selecting a country of origin from the "Select Codepage" dialog. This information is saved in OS2.INI, and is used by the PicIchg API if present.

The text strings in the output metafile from a PIF to metafile conversion are by default expressed in an ASCII codepage. Optionally they can be expressed in the EBCDIC codepage originally used in the PIF. This option is selected by information also stored in OS2.INI by the "Select Codepage" dialog of the Convert Picture File utility. This option is provided so that fonts produced by PicIchg, which retain their EBCDIC codepage, can be referenced.

PicPrint – Picture Print

On page 5-5, remove the information about the **MAP** parameter. This parameter is not supported.

© Copyright IBM Corp. 1990

Chapter 6. Profile Function Calls

Summary of Changes

New	Updated	Section Title
	√	PrfQueryProgramTitles
	√	PrfReset

PrfQueryProgramTitles — Query Program Titles

On page 6-21, in the **Group** parameter, change **SGH_GROUP** to **SGH_ROOT.**

PrfReset — Reset Presentation Manager

On page 6-24, change the **Profile** parameter description to the following.

Profile *(PRFPROFILE)* — input
Profile-names structure.

This contains the name(s) of the file(s) to be used as the new Presentation Manager profile files. Any valid file name(s) can be used. A name that is not already fully qualified is taken to refer to the current directory.

If the user profile file does not exist, a new file is created.

The name of the system profile cannot be changed. It must be the name of the current system profile as returned by PrfQueryProfile.

© Copyright IBM Corp. 1990

Chapter 7. Spooler Function Calls

Summary of Changes

New	Updated	Section Title
√		DosPrintDestAdd
√		DosPrintDestControl
√		DosPrintDestDel
√		DosPrintDestEnum
√		DosPrintDestGetInfo
√		DosPrintDestSetInfo
√		DosPrintJobContinue
√		DosPrintJobDel
√		DosPrintJobEnum
√		DosPrintJobGetId
√		DosPrintJobGetInfo
√		DosPrintJobPause
√		DosPrintSetInfo
√		DosPrintQAdd
√		DosPrintQContinue
√		DosPrintQDel
√		DosPrintQEnum
√		DosPrintQGetInfo
√		DosPrintQPause
√		DosPrintQPurge
√		DosPrintQSetInfo

© Copyright IBM Corp. 1990

DosPrintDestAdd — Add Print Destination

This call establishes a print destination on the local workstation or a remote server.

DosPrintDestAdd (ComputerName, Level, Buf, Length, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where destination is to be added.

A NULL string specifies the local workstation.

Level *(USHORT)* — input
Level of detail provided.

This must be 3.

Buf *(PBYTE)* — input
Data structure.

Length *(COUNT2B)* — input
Size, in bytes, of data structure.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_NAME (123)	The computer name is invalid.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_DestExists (2153)	The printer destination already exists.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_DestInvalidState (2162)	This operation cannot be performed on the print destination.
NERR_SpoolNoMemory (2165)	A spooler memory allocation failure occurred.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

The result of this call is the creation of a new printer definition.

The printer is set up to print on the logical address defined by **logaddr** in *PRDINFO3*. If **logaddr** is NULL, the printer definition is created but is not connected to any logical address. In this case no printing can occur on that printer or from any print queue connected only to that printer. If a logical address is specified, it must already be defined in the PM_SPOOLER_PORTS section of the initialization file.

Programming Note: To change the connection between a printer and a port, use DosPrintDestSetInfo.

The maximum length for a destination name is 32 characters under OS/2 Presentation Manager. The use of a longer name results in ERROR_INVALID_NAME (123).

All device drivers and queues specified with the printer must already be defined to the spooler.

To add a remote print destination requires administrator privilege.

DosPrintDestControl — Control Print Destination

This call cancels, holds, continues, or restarts a print destination.

DosPrintDestControl (ComputerName, PrinterName, Control, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where destination is to be controlled.

A NULL string specifies the local workstation.

PrinterName *(STRL)* — input
Name of print destination.

Control *(BIT16)* — input
Operation to perform.

PRD_DELETE	Delete current print job
PRD_PAUSE	Pause printing
PRD_CONT	Continue paused print job
PRD_RESTART	Restart print job.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_DestNotFound (2152)	The printer destination cannot be found.
NERR_DestIdle (2158)	This printer destination is idle and cannot accept control operations.
NERR_DestInvalidOp (2159)	This printer destination request contains an invalid control function.
NERR_ProcNoRespond (2160)	The queue processor is not responding.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

A paused print destination cannot accept new print jobs.

If PRD_DELETE is attempted when there is no current print job, NERR_DestIdle (2158) is returned.

To control jobs on a remote server requires administrator privilege.

DosPrintDestDel – Delete Print Destination

This call deletes a print destination.

DosPrintDestDel (ComputerName, PrinterName, *rc*)

Parameters

ComputerName *(STRL)* – input
Name of computer where destination is to be deleted.

A NULL string specifies the local workstation.

PrinterName *(STRL)* – input
Name of Print Destination.

rc *(SPLERR)* – return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_DestNotFound (2152)	The printer destination cannot be found.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_DestInvalidState (2162)	This operation cannot be performed on the print destination.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

If the destination is currently printing a job, DosPrintDestDel fails and returns NERR_DestInvalidState (2162).

To delete a destination on a remote server requires administrator privilege.

DosPrintDestEnum — Enumerate Print Destination

This call lists print destinations on a server, optionally supplying status information.

DosPrintDestEnum (ComputerName, Level, Buf, Length, Returned, Total, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where destinations are to be listed.

A NULL string specifies the local workstation.

Level *(USHORT)* — input
Level of detail required.

This must be 0, 1, 2, or 3.

Buf *(PBYTE)* — output
Data structure.

Length *(COUNT2B)* — input
Size, in bytes, of data structure.

Returned *(COUNT2)* — output
Number of entries returned.

Total *(COUNT2)* — output
Number of entries available.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

The buffer contents on return are:

Level	Buffer Contents
0	An array of printer ports, each of type CHAR*PDLEN+1
1	An array of *PRDINFO* structures.
2	An array of printer names, each of type STRL
3	An array of *PRDINFO3* structures.

If no job is printing on a print destination, bits 2 – 11 of **status** in the *PRDINFO3* data structure, or bits 2 – 8 of **status** in the *PRDINFO* data structure are meaningless.

DosPrintDestGetInfo – Get Print Destination Information

This call retrieves information about a print destination.

DosPrintDestGetInfo (ComputerName, PrinterName, Level, Buf, Length, Needed, *rc*)

Parameters

ComputerName *(STRL)* – input
Name of computer where destination is to be queried.

A NULL string specifies the local workstation.

PrinterName *(STRL)* – input
Name of Print Destination.

This can specify a printer name or a port name. If **Level** is 0 or 1, it must be a logical address. If **Level** is 2 or 3, it must be a printer name.

Level *(USHORT)* – input
Level of detail required.

This must be 0, 1, 2, or 3.

Buf *(PBYTE)* – output
Data structure.

Length *(COUNT2B)* – input
Size, in bytes, of data structure.

Needed *(COUNT2B)* – output
Number of bytes of information available.

rc *(SPLERR)* – return

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_DestNotFound (2152)	The printer destination cannot be found.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

The buffer contents on return are:

Level	Buffer Contents
0	A logical address.
1	A *PRDINFO* structure.
2	A printer name.
3	A *PRDINFO3* structure.

If no job is printing on the print destination, bits 2 – 11 of **status** in the *PRDINFO3* data structure, or bits 2 – 8 of **status** in the *PRDINFO* structure, are meaningless.

DosPrintDestSetInfo — Set Print Destination Information

This call modifies the configuration of a print destination.

DosPrintDestSetInfo (ComputerName, PrinterName, Level, Buf, Length, ParmNum, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where destination is to be modified.

A NULL string specifies the local workstation.

PrinterName *(STRL)* — input
Name of Print Destination.

Level *(USHORT)* — input
Level of detail.

The level of detail provided in *PRDINFO3*. This must be 3.

Buf *(PBYTE)* — input
Data structure.

Length *(COUNT2B)* — input
Size, in bytes, of data structure.

ParmNum *(USHORT)* — input
Parameter number.

Specifies either that the entire *PRDINFO3* structure is to be modified, or that one specific parameter only is to be modified. If **ParmNum** is 0, **Buf** must contain a complete *PRDINFO3* structure. Otherwise, **ParmNum** must contain the position value corresponding to the parameter to be modified:

Parameter	Constant (Value)
logaddr	PRD_LOGADDR_PARMNUM (3)
comment	PRD_COMMENT_PARMNUM (7)
drivers	PRD_DRIVERS_PARMNUM (8)
timeout	PRD_TIMEOUT_PARMNUM (10)

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_DestNotFound (2152)	The printer destination cannot be found.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.

NERR_DestInvalidState (2162)	This operation cannot be performed on the print destination.
NERR_SpoolNoMemory (2165)	A spooler memory allocation failure occurred.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

This call allows modification of a printer and its connection to a logical address. To disconnect a printer from a port, use **Level**=3, **ParmNum**=2, and **Buf** is NULL.

To modify a printer destination on a remote server requires administrator privilege.

DosPrintJobContinue — Continue Print Job

This call continues a paused print job.

DosPrintJobContinue (ComputerName, Job, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where job is to be continued.

A NULL string specifies the local workstation.

Job *(IDENTITY)* — input
Job identification number.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_JobInvalidState (2164)	This operation cannot be performed on the print job in its current state.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

Any job can be released by a user with administrator privilege.

A job created locally can be released locally regardless of user privilege level, but it can be released remotely only by a user with administrator privilege.

A remote job can be released by a user without administrator privilege only if the user identification of the person initiating the request is the same as the user identification of the person who created the job.

DosPrintJobDel — Delete Print Job

This call deletes a job from a print queue.

DosPrintJobDel (ComputerName, Job, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where job is to be deleted.

A NULL string specifies the local workstation.

Job *(IDENTITY)* — input
Job identification number.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_ProcNoRespond (2160)	The queue processor is not responding.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

It is possible to delete a job that is currently printing.

If the print queue on which the print job is submitted is pending deletion (following a DosPrintQDel call), and the print job is the last in the queue, this call has the additional effect of deleting the queue.

A user with administrator privilege can delete any job.

A job created locally can be deleted locally regardless of user privilege level, but can be deleted remotely only by an administrator.

A remote job can be deleted by a user without administrator privilege only if the username of the person initiating the request is the same as the username of the person who created the job.

DosPrintJobEnum — Enumerate Print Job

This call lists the jobs in a print queue, optionally supplying status information on each job.

DosPrintJobEnum (ComputerName, QueueName, Level, Buf, Length, Returned, Total, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where jobs are to be listed.

A NULL string specifies the local workstation.

QueueName *(STRL)* — input
Queue name.

Level *(USHORT)* — input
Level of detail required.

This must be 0, 1, or 2.

Buf *(PBYTE)* — output
Data structure.

Length *(COUNT2B)* — input
Size, in bytes, of data structure.

Returned *(COUNT2)* — output
Number of entries returned.

Total *(COUNT2)* — output
Number of entries available.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_INVALID_NAME (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

The buffer contents on return are:

Level	Buffer Contents
0	An array containing a **jobId** for each of **Returned** jobs.
1	An array containing a *PRJINFO* structure for each of **Returned** jobs.
2	An array containing a *PRJINFO2* structure for each of **Returned** jobs.

DosPrintJobGetId — Get Print Job ID

This call retrieves information about a remote print job.

DosPrintJobGetId (File, Info, Length, *rc*)

Parameters

File *(HFILE)* — input
Handle of redirected print device.

Info *(PRIDINFO)* — output
Data structure.

Length *(COUNT2B)* — input
Size, in bytes, of data structure.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_INVALID_HANDLE (6)	An invalid handle is specified.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
NERR_DevNotRedirected (2107)	The device is not connected.
NERR_BufTooSmall (2123)	The API return buffer is too small.

Remarks

This call is available only for redirected devices and remote queues. It is not available for the local spooler. If it is called when the LAN program is not installed, ERROR_NOT_SUPPORTED (50) is returned.

This call is not available on file handles for local jobs. It is available only for redirected devices and remote queues. This applies even when using the LAN Manager spooler.

The buffer contents on return are a *PRIDINFO* structure.

DosPrintJobGetInfo – Get Print Job Information

This call retrieves information about a print job.

DosPrintJobGetInfo (ComputerName, Job, Level, Buf, Length, Needed, *rc*)

Parameters

ComputerName *(STRL)* – input
Name of computer where print job is to be queried.

A NULL string specifies the local workstation.

Job *(IDENTITY)* – input
Job identification number.

Level *(USHORT)* – input
Level of detail required.

This must be 0, 1, 2, or 3.

Buf *(PBYTE)* – output
Data structure.

Length *(COUNT2B)* – input
Size, in bytes, of data structure.

Needed *(COUNT2B)* – output
Size in bytes of available information.

rc *(SPLERR)* – return

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

The buffer contents on return are:

Level	Buffer Contents
0	The job identifier
1	A *PRJINFO* structure, with variable information, up to the **Length** of **Buf**
2	A *PRJINFO2* structure, with variable information, up to the **Length** of **Buf**
3	A *PRJINFO3* structure, with variable information, up to the **Length** of **Buf**.

DosPrintJobPause — Pause Print Job

This call pauses a job in a print queue.

```
DosPrintJobPause (ComputerName, Job, rc)
```

Parameters

ComputerName *(STRL)* — input
Name of computer where job is to be paused.

A NULL string specifies the local workstation.

Job *(IDENTITY)* — input
Job identification number.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not installed.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_JobInvalidState (2164)	This operation cannot be performed on the print job in its current state.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

If the job is already printing when the call is made, NERR_JobInvalidState (2164) is returned.

A user with administrator privilege can pause any job.

A job created locally can be paused locally regardless of user privilege level, but can be paused remotely only by an administrator.

A remote job can be paused by a user without administrator privilege only if the username of the person initiating the request is the same as the username of the person who created the job.

DosPrintJobSetInfo — Set Print Job Information

This call modifies the instructions for a print job.

DosPrintJobSetInfo (ComputerName, Job, Level, Buf, Length, ParmNum, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where job is to be modified.

A NULL string specifies the local workstation.

Job *(IDENTITY)* — input
Job identification number.

Level *(USHORT)* — input
Level of detail required.

This must be 1 or 3.

Buf *(PBYTE)* — input
Data structure.

This is either a *PRJINFO* structure (if **Level** is 1) or a *PRJINFO3* structure (if **Level** is 3).

Length *(COUNT2B)* — input
Size, in bytes, of data structure.

ParmNum *(USHORT)* — input
Parameter number.

Specifies either that the entire job structure is to be modified, or that one specified parameter only is to be modified.

If **ParmNum** is 0 and **Level** is 1, **Buf** must contain a complete *PRJINFO* structure. If **ParmNum** is 0 and **Level** is 3, **Buf** must contain a complete *PRJINFO3* structure. Otherwise, **ParmNum** must contain the position value corresponding to the parameter to be modified:

Parameter	Constant (Value)
notifyname	PRJ_NOTIFYNAME_PARMNUM (3)
datatype	PRJ_DATATYPE_PARMNUM (4)
parms	PRJ_PARMS_PARMNUM (5)
position	PRJ_POSITION_PARMNUM (6)
comment	PRJ_COMMENT_PARMNUM (11)
document	PRJ_DOCUMENT_PARMNUM (12)
priority	PRJ_PRIORITY_PARMNUM (14)
qprocparms	PRJ_PROCPARMS_PARMNUM (16)
driverdata	PRJ_DRIVERDATA_PARMNUM (18)

position must be given the appropriate value:

Value	Change
0	No change
1	Move to first place

>1	Move to this position, or if the specified value is greater than the number of jobs in the queue, move to the end of the queue.

rc *(SPLERR)* – return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_JobNotFound (2151)	The print job does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_JobInvalidState (2164)	This operation cannot be performed on the print job in its current state.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

The job priority is changed, if necessary, to the priority of the next job after the new position. If the spooler is restarted, the order in which jobs are put on the queue depends on the priority and age of the job. This order may be different from the order following the DosPrintJobSetInfo call.

Users without administrator privilege can use DosPrintJobSetInfo only for jobs created when the same user name was logged on. They can move jobs backwards only, and cannot increase the job priority above the queue priority.

A job created locally has no associated user name, and any user can set information locally for the job. Only an administrator can set information for a job on a remote server.

DosPrintQAdd – Add Print Queue

This call creates a new print queue on the local workstation or on a remote server. A remote server setup requires the LAN Requester and Server software.

DosPrintQAdd (ComputerName, Level, Buf, Length, *rc*)

Parameters

ComputerName *(STRL)* – input
Name of computer where queue is to be created.

A NULL string specifies a local workstation.

Level *(USHORT)* – input
Level of detail provided.

This must be 1 or 3.

Buf *(PBYTE)* – input
Data structure.

If **Level** is 1, this is a *PRQINFO* structure; if **Level** is 3, this is a *PRQINFO3* structure.

Length *(COUNT2B)* – input
Size, in bytes, of data structure.

rc *(SPLERR)* – return

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_NAME (123)	The computer name is invalid.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_QExists (2154)	The printer queue already exists.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_SpoolNoMemory (2165)	A spooler memory allocation failure occurred.
NERR_ProcNotFound (2168)	The queue processor is not installed.
NERR_InvalidDevice (2294)	The requested device is invalid.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

To create a queue on a remote server requires administrator privilege.

Applications wanting to create print queues should use the level 3 call with a *PRQINFO3* data structure. Level 1 calls are provided only for compatibility, and should not be used by new programs.

The following fields are required in *PRQINFO3*:

name
priority
starttime
untiltime
sepfile
parms
printerdestinations
drivername
driverdata.

The following fields in the data structure can be set but will have no effect unless the queue is being created on a remote server that has the LAN Server program installed :

starttime
untiltime
sepfile.

The maximum length of a queue name under OS/2 Presentation Manager is 9 characters (including a terminating NULL). A longer queue name results in the error ERROR_INVALID_NAME (123).

If a queue of the name specified in **name** already exists on **ComputerName**, the call fails unless the queue is marked for deletion. In this case, the queue is not deleted, and the creation fields are used to perform a DosPrintQAdd call on the queue.

If **printerdestinations** is NULL, the queue is created but not connected to any printer.

The queue that is created has a status of PRQ3_PENDING even if the queue is not connected to a printer.

drivername can be a NULL string, in which case **driverdata** is ignored. Otherwise **drivername** must refer to the name of a device driver that is already defined in the initialization file (for example, "IBM4201").

DosPrintQContinue — Continue Print Queue

This call continues a paused print queue.

DosPrintQContinue (ComputerName, QueueName, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where queue is to be continued.

A NULL string specifies the local workstation.

QueueName *(STRL)* — input
Queue name.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

This call continues a queue that has been paused by a DosPrintQPause call, or disabled by an error on the queue. It does not affect an active print queue.

To continue a queue on a remote server requires administrator privilege on the remote server.

DosPrintQDel – Delete Print Queue

This call deletes a print queue from the spooler.

```
DosPrintQDel (ComputerName, QueueName, rc)
```

Parameters

ComputerName *(STRL)* – input
Name of computer where queue is to be deleted.

A NULL string specifies the local workstation.

QueueName *(STRL)* – input
Queue name.

rc *(SPLERR)* – return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_QInvalidState (2163)	This operation cannot be performed on the print queue.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

If there are print jobs in the queue, DosPrintQDel marks the queue PRQ3_PENDING. No further jobs can then be added to the queue, which is deleted when all jobs are printed. A queue marked PRQ3_PENDING cannot be paused and continued, but jobs in the queue can be paused, restarted, and repeated.

To delete a queue on a remote server requires administrator privilege on the remote server.

DosPrintQEnum – Enumerate Print Queue

This call lists print queues on the local workstation or on a remote server, optionally supplying additional information.

DosPrintQEnum (ComputerName, Level, Buf, Length, Returned, Total, *rc*)

Parameters

ComputerName *(STRL)* – input
Name of computer where queues are to be listed.

A NULL string specifies the local workstation.

Level *(USHORT)* – input
Level of detail.

The level of detail required. This must be 0, 1, 2, 3, or 4.

Buf *(PBYTE)* – output
Data structure.

Length *(COUNT2B)* – input
Size, in bytes, of data structure.

Returned *(COUNT2)* – output
Number of entries returned.

Total *(COUNT2)* – output
Total number of entries available.

rc *(SPLERR)* – return

NO_ERROR (0)	No errors occurred.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

The buffer contents on return are:

Level	Buffer Contents
0	An array of queue names, each of type *CHAR*QNLEN+1*
1	An array of *PRQINFO* structures
2	An array of *PRQINFO* structures in which each *PRQINFO* structure is followed by an array of *PRJINFO* structures, one for each job in the queue
3	An array of *PRQINFO3* structures

4 An array of **Returned** *PRQINFO3* structures in which each *PRQINFO3* structure is followed by an array of *PRJINFO2* structures, one for each job in the queue.

DosPrintQGetInfo — Get Print Queue Information

This call supplies information about a print queue, and, optionally, about the jobs in it.

DosPrintQGetInfo (ComputerName, QueueName, Level, Buf, Length, Needed, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where queue is to be queried.

A NULL string specifies the local workstation.

QueueName *(STRL)* — input
Queue name.

Level *(USHORT)* — input
Level of detail required.

This must be 0, 1, 2, 3, or 4.

Buf *(PBYTE)* — input
Data structure.

Length *(COUNT2B)* — input
Size, in bytes, of data structure.

Needed *(COUNT2B)* — output
Size, in bytes, of available information.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
ERROR_MORE_DATA (234)	Additional data is available.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

The buffer contents on return are:

Level	Buffer Contents
0	The queue name.
1	A single *PRQINFO* structure, with associated variable information up to **Length**.

2 A *PRQINFO* structure, with associated variable information, and an array of *PRJINFO* structures, up to **Length**.

3 A *PRQINFO3* structure, with associated variable information up to **Length**.

4 A *PRQINFO3* structure, with associated variable information, and an array of *PRJINFO2* structures, one for each job in the queue, up to **Length**.

If **Level** is 1, 2, 3, or 4, and **Buf** cannot hold the entire *PRQINFO* or *PRQINFO3* structure, DosPrintQGetInfo returns NERR_BufTooSmall (2123).

If **Level** is 2, and **Buf** cannot hold all the available *PRJINFO* structures, DosPrintQGetInfo returns ERROR_MORE_DATA (234).

If **Level** is 4, and **Buf** cannot hold all the available *PRJINFO2* structures, DosPrintQGetInfo returns ERROR_MORE_DATA (234).

To obtain the size of buffer required, call DosPrintQGetInfo with the required value of **Level** and a NULL buffer. The number of bytes required is returned in **Needed**.

DosPrintQPause — Pause Print Queue

This call pauses a print queue.

DosPrintQPause (ComputerName, QueueName, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where queue is to be paused.

A NULL string specifies the local workstation.

QueueName *(STRL)* — input
Queue name.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

This call suspends processing of all print jobs except for a job currently printing. Print jobs can be submitted to a paused queue, but no jobs will be printed until the queue is continued by a DosPrintQPause call.

To pause a remote queue requires administrator privilege on the remote server.

DosPrintQPurge — Purge Print Queue

This call removes all jobs, except any currently printing, from a print queue.

DosPrintQPurge (ComputerName, QueueName, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where queue is to be purged.

A NULL string specifies the local workstation.

QueueName *(STRL)* — input
Queue name.

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.
ERROR_INVALID_PARAMETER (87)	An invalid parameter was specified.
NERR_NetNotStarted (2102)	The network program is not started.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

A print job that is printing is not affected by this call.

If a print queue is pending deletion when this call is made, the queue is deleted when the job that is currently printing ends.

To purge a remote queue requires administrator privilege on the remote server.

DosPrintQSetInfo — Set Print Queue Information

This call modifies the configuration of a print queue.

DosPrintQSetInfo (ComputerName, QueueName, Level, Buf, Length, ParmNum, *rc*)

Parameters

ComputerName *(STRL)* — input
Name of computer where queue is to be modified.

A NULL string specifies the local workstation.

QueueName *(STRL)* — input
Queue name.

Level *(USHORT)* — input
Level of detail required.

This must be 1 or 3.

Buf *(PBYTE)* — input
Data structure.

Length *(COUNT2B)* — input
Size, in bytes, of data structure.

ParmNum *(USHORT)* — input
Parameter number.

Specifies either that the entire *PRQINFO3* structure is to be modified, or that one specified parameter only is to be modified.

If **ParmNum** is 0, **Buf** must contain a complete *PRQINFO3* structure. Otherwise, **ParmNum** must contain the position value corresponding to the parameter to be modified:

Parameter	Constant (Value)
priority	PRQ_PRIORITY_PARMNUM (2)
starttime	PRQ_STARTTIME_PARMNUM (3)
untiltime	PRQ_UNTILTIME_PARMNUM (4)
sepfile	PRQ_SEPARATOR_PARMNUM (5)
prproc	PRQ_PROCESSOR_PARMNUM (6)
parms	PRQ_PARMS_PARMNUM (8)
comment	PRQ_COMMENT_PARMNUM (9)
printerdestinations	PRQ_PRINTERS_PARMNUM (12)
drivername	PRQ_DRIVERNAME_PARMNUM (13)
driverdata	PRQ_DRIVERDATA_PARMNUM (14)

rc *(SPLERR)* — return

NO_ERROR (0)	No errors occurred.
ERROR_ACCESS_DENIED (5)	Access is denied.
ERROR_NOT_SUPPORTED (50)	This request is not supported by the network.
ERROR_BAD_NETPATH (53)	The network path cannot be located.

ERROR_INVALID_PARAMETER (87)	An invalid parameter is specified.
ERROR_INVALID_LEVEL (124)	The level parameter is invalid.
NERR_NetNotStarted (2102)	The network program is not installed.
NERR_BufTooSmall (2123)	The API return buffer is too small.
NERR_QNotFound (2150)	The printer queue does not exist.
NERR_SpoolerNotLoaded (2161)	The spooler is not running.
NERR_SpoolNoMemory (2165)	A spooler memory allocation failure occurred.
NERR_InvalidDevice (2294)	The requested device is invalid.
NERR_InvalidComputer (2351)	The computer name is invalid.

Remarks

If the **priority** field in *PRQINFO3* or *PRQINFO* is set to PRQ_NO_PRIORITY, the queue priority is not changed.

Chapter 8. Advanced Video Function Calls

Summary of Changes

New	Updated	Section Title
	√	VioQueryFonts
	√	VioSetDeviceCellSize

VioQueryFonts — Vio Query Fonts

On page 8-11, add the following to the **Remarks** section:

A default font, suitable for use with VIO, is provided automatically for each VIO-supported cell size; default fonts are not enumerated by VioQueryFonts. Valid cell sizes for the device can be found using DevQueryCaps (CAPS_CHAR_WIDTH with CAPS_CHAR_HEIGHT and CAPS_SMALL_CHAR_WIDTH with CAPS_SMALL_CHAR_HEIGHT if the device supports two sizes). Devices that support more than two cell sizes also support the DEVESC_QUERYVIOCELLSIZES escape (see DevEscape).

VioSetDeviceCellSize — Vio Set Device Cell Size

On page 8-13, replace the table in the Remarks section with:

Adapter	Cell Sizes
CGA	8 x 8
EGA	8 x 8, 8 x 10, 8 x 12, 8 x 14 (default), 8 x 16, 8 x 18
VGA	8 x 8, 8 x 10, 8 x 12, 8 x 14 (default), 8 x 16, 8 x 18
8514/A	6 x 14, 7 x 12, 7 x 15, 7 x 25, 8 x 8, 8 x 12, 8 x 14, 8 x 17 (default), 12 x 16, 12 x 20, 12 x 22, 12 x 30
XGA	6 x 10, 6 x 14, 7 x 15, 7 x 25, 8 x 8, 8 x 10, 8 x 12, 8 x 14, 8 x 16, 8 x 18, 10 x 18, 12 x 16, 12 x 20, 12 x 22, 12 x 30. (8 x 14 is the default for a 640 x 480 resolution monitor, and 12 x 22 is the default for a 1024 x 768 resolution monitor.)

© Copyright IBM Corp. 1990

Chapter 9. Window Function Calls

Summary of Changes

New	Updated	Section Title
	√	Message Queues and Window Calls
	√	WinAddSwitchEntry
	√	WinAllocMem
	√	WinBeginPaint
	√	WinCalcFrameRect
	√	WinChangeSwitchEntry
	√	WinCopyRect
	√	WinCreateCursor
	√	WinCreateHeap
	√	WinCreateMsgQueue
	√	WinCreateStdWindow
	√	WinCreateSwitchEntry
	√	WinDestroyWindow
	√	WinDrawBitmap
	√	WinDrawBorder
	√	WinDrawText
	√	WinEnumDlgItem
	√	WinEqualRect
	√	WinFillRect
	√	WinFreeMem
	√	WinGetErrorInfo
	√	WinInflateRect
	√	WinInitialize
	√	WinIntersectRect
	√	WinInvalidateRect
	√	WinInvertRect
	√	WinIsRectEmpty
	√	WinLoadDlg
	√	WinLockVisRegions
	√	WinMessageBox
	√	WinOffsetRect
	√	WinPtInRect
	√	WinQueryQueueStatus
	√	WinQuerySysValue
	√	WinQueryUpdateRect

© Copyright IBM Corp. 1990

New	Updated	Section Title
	√	WinQueryWindowRect
	√	WinQueryWindowUshort
	√	WinReallocMem
	√	WinRegisterClass
	√	WinScrollWindow
	√	WinSetClassMsgInterest
	√	WinSetClipbrdData
	√	WinSetHook
	√	WinSetMsgInterest
	√	WinSetRect
	√	WinSetRectEmpty
	√	WinSetSysValue
	√	WinSubtractRect
	√	WinTractRect
	√	WinUnionRect
	√	WinValidateRect

Message Queues and Window Calls

The following Presentation Manager window calls require a message queue. This means that, before you issue one of these calls, you must call WinCreateMsgQueue from the same thread.

This list supersedes the statement on message queues that is given in the description of some calls in the *Presentation Manager Programming Reference Volume 1*.

WinAssociateHelpInstance
WinBeginEnumWindows
WinBeginPaint
WinCalcFrameRect
WinCallMsgFilter
WinCancelShutdown
WinCatch
WinCloseClipbrd
WinCopyAccelTable
WinCreateAccelTable
WinCreateCursor
WinCreateDlg
WinCreateFrameControls
WinCreateMenu
WinCreatePointer
WinCreatePointerIndirect
WinCreateStdWindow
WinCreateWindow
WinDdeInitiate
WinDdePostMsg
WinDdeRespond
WinDefAVioWindowProc
WinDefDlgProc
WinDefWindowProc
WinDeleteLibrary
WinDeleteProcedure
WinDestroyAccelTable
WinDestroyCursor
WinDestroyMsgQueue
WinDestroyPointer
WinDestroyWindow
WinDismissDlg
WinDispatchMsg
WinDlgBox
WinDrawBorder
WinEmptyClipbrd
WinEnablePhysInput
WinEnableWindow

WinEnableWindowUpdate
WinEndEnumWindows
WinEndPaint
WinEnumClipbrdFmts
WinEnumDlgItem
WinExcludeUpdateRegion
WinFlashWindow
WinFocusChange
WinGetClipPS
WinGetDlgMsg
WinGetKeyState
WinGetMinPosition
WinGetMsg
WinGetNextWindow
WinGetPhysKeyState
WinGetPS
WinGetScreenPS
WinInSendMsg
WinInstStartApp
WinInvalidateRect
WinInvalidateRegion
WinIsChild
WinIsThreadActive
WinIsWindowEnabled
WinIsWindowShowing
WinIsWindowVisible
WinLoadAccelTable
WinLoadDlg
WinLoadLibrary
WinLoadMenu
WinLoadProcedure
WinLockVisRegions
WinLockWindowUpdate
WinMapDlgPoints
WinMapWindowPoints
WinMessageBox
WinMsgMuxSemWait
WinMsgSemWait
WinMultWindowFromIDs
WinOpenClipbrd
WinOpenWindowDC
WinPeekMsg
WinProcessDlg
WinQueryAccelTable
WinQueryActiveWindow
WinQueryCapture
WinQueryClassInfo
WinQueryClassName
WinQueryClipbrdData
WinQueryClipbrdFmtInfo
WinQueryClipbrdOwner
WinQueryClipbrdViewer
WinQueryCp
WinQueryCursorInfo
WinQueryDesktopWindow
WinQueryDlgItemShort
WinQueryDlgItemText
WinQueryDlgItemTextLength
WinQueryFocus
WinQueryMsgPos
WinQueryMsgTime
WinQueryObjectWindow
WinQueryPointer
WinQueryPointerInfo
WinQueryPointerPos
WinQueryPresParam
WinQueryQueueStatus
WinQuerySysModalWindow
WinQuerySysPointer
WinQueryUpdateRect
WinQueryUpdateRegion
WinQueryWindow
WinQueryWindowDC
WinQueryWindowPos
WinQueryWindowPtr
WinQueryWindowRect
WinQueryWindowText
WinQueryWindowTextLength
WinQueryWindowULong
WinQueryWindowUShort
WinRegisterClass
WinRegisterUserDatatype
WinRegisterUserMsg
WinRegisterWindowDestroy
WinReleaseHook
WinReleasePS
WinRemovePresParam
WinScrollWindow
WinSendDlgItemMsg
WinSendMsg
WinSetAccelTable
WinSetActiveWindow
WinSetCapture
WinSetClassMsgInterest
WinSetClipbrdData
WinSetClipbrdOwner
WinSetClipbrdViewer
WinSetCp
WinSetDlgItemShort
WinSetDlgItemText
WinSetFocus
WinSetHook
WinSetMsgInterest
WinSetMsgMode
WinSetMultWindowPos
WinSetOwner
WinSetParent
WinSetPointer
WinSetPointerPos
WinSetPresParam
WinSetSynchroMode
WinSetSysColors

WinSetSysModalWindow
WinSetWindowBits
WinSetWindowPos
WinSetWindowPtr
WinSetWindowText
WinSetWindowULong
WinSetWindowUShort
WinShowCursor
WinShowPointer
WinShowTrackRect
WinShowWindow
WinStartTimer
WinStopTimer
WinSubclassWindow
WinSubstituteStrings
WinTerminateApp
WinThrow
WinTrackRect
WinTranslateAccel
WinUpdateWindow
WinValidateRect
WinValidateRegion
WinWaitMsg
WinWindowFromDC
WinWindowFromID
WinWindowFromPoint

WinAddSwitchEntry — Add Switch Entry

On page 9-12, under the **SwitchData** parameter, note that leading and trailing blanks are not removed from the title.

WinAllocMem — Allocate Heap Space

On page 9-14, change the first paragraph of the Remarks section to:

This call returns the 16-bit offset from the start of the segment containing the **Heap** of the allocated memory object. The memory offset returned is aligned on a *ULONG* boundary.

WinBeginPaint — Begin Paint

On page 9-18, change the Programming Note for the **Rect** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinCalcFrameRect — Calculate Frame Rectangle

On page 9-20, change the Programming Note for the **Rect** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinChangeSwitchEntry — Change Switch Entry

On page 9-24, add the following to the **SwitchData** parameter description:

If the **process** field of the *SWCNTRL* structure is NULL, the current process ID is used.

If the **session** field of the *SWCNTRL* structure is NULL, the current session ID is used.

WinCopyRect — Copy Rectangle

On page 9-28, change the Programming Notes for the **Dest** and **Src** parameters to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinCreateCursor — Create Cursor

On page 9-33, change the Programming Note for the **Clip** parameter to:

The cursor is always clipped to the window rectangle, even if part of **Clip** is outside it.

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinCreateHeap — Create Heap

On page 9-42, add the following to the programming note:

This is particularly important for the command line and environment parameters (**argv** and **envp**) passed to the main routine of a C-language application. These parameters are pointers to structures that are allocated using C-language memory allocation functions, and will become invalid if WinCreateHeap is used to create a heap in the automatic data segment. In this case, the application must either process these parameters before creating a heap, or use the DosGetInfoSeg call to access the command line and environment.

WinCreateMsgQueue — Create Message Queue

On page 9-46, add the following to the list of possible returns from WinGetLastError:

PMERR_NOT_IN_A_PM_SESSION
PMERR_QUEUE_ALREADY_EXISTS

WinCreateStdWindow — Create Standard Window

On page 9-49, change the last sentence of the description of the **Style** parameter to:

The interpretation of the parameters is affected by the use of all the styles, except for WS_MINIMIZED and WS_MAXIMIZED. These two styles are ignored if they are specified.

WinCreateSwitchEntry — Create Switch Entry

On page 9-51, add the following to the **SwitchData** parameter description:

If the **session** field of the *SWCNTRL* structure is NULL, the current session ID is used.

WinDestroyWindow — Destroy Window

On page 9-73, add the following paragraph to the Remarks section:

If a menu window is being destroyed, any bit maps associated with the menu are not deleted. They will be deleted automatically when the application terminates.

WinDrawBitmap — Draw Bit Map

On page 9-79, change the Programming Note for the **Src** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinDrawBorder — Draw Border

On page 9-81, change the Programming Note for the **Rectangle** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinDrawText — Draw Text

On page 9-84, change the Programming Note for the **Rectangle** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

On page 9-84, under the **Cmd** parameter description, remove the line:

DT_WORDBREAK can only be specified with DT_TOP and DT_LEFT.

WinEnumDlgItem — Enumerate Dialog Item

On page 9-93, add the following to the **hwnd** parameter description:

NULL can be specified if **Code** is EDI_FIRSTTABITEM or EDI_LASTTABITEM.

WinEqualRect — Equal Rectangle

On page 9-94, change the Programming Notes for the **Rect1** and **Rect2** parameters to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

On page 9-94, add the following:

Remarks

If both rectangles are empty (for example, **ytop** is equal to **ybottom** or **xright** is equal to **xleft**), they are considered equal even if the actual coordinate values are different.

WinFillRect — Fill Rectangle

On page 9-96, change the Programming Note for the **Rect** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinFreeMem — Free Memory on Heap

On page 9-102, change the first paragraph of the Remarks section to:

The value of the **Mem** parameter must have been returned by either the WinAllocMem or WinReallocMem call.

WinGetErrorInfo — Get Error Information

On page 9-108, delete the third paragraph of the Remarks section, beginning 'Like the WinGetLastError...'.

WinInflateRect — Inflate Rectangle

On page 9-119, change the Programming Note for the **Rect** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinInitialize — Initialize

On page 9-120, delete the example given in the **hab** parameter.

WinIntersectRect — Intersect Rectangle

On page 9-124, change the Programming Notes for the **Dest**, **Rect1**, and **Rect2** parameters to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinInvalidateRect — Invalidate Rectangle

On page 9-125, change the Programming Note for the **Prc** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinInvertRect — Invert Rectangle

On page 9-127, change the Programming Note for the **Rect** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinIsRectEmpty — Is Rectangle Empty

On page 9-129, change the Programming Note for the **prc** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinLoadDlg — Load Dialog

On page 9-136, change the first paragraph of the Remarks section to:

Unless window style WS_VISIBLE is specified for the dialog window in the DIALOG statement within the dialog template, the dialog window is created as an invisible window

WinLockVisRegions — Lock Visible Regions

On page 9-145, add the following to the Remarks section:

Programming Note: Locking the visible regions does not prevent painting of a window by another process.

WinMessageBox — Message Box

On page 9-154, replace the eighth paragraph in the Remarks section, beginning 'Text is word-wrapped...' with:

Text is wrapped at word boundaries (spaces). If a word is too big to fit on one line, the start of the word is not wrapped to the next line, but stays adjacent to the text it follows, and the word is split at the box boundary.

WinOffsetRect — Offset Rectangle

On page 9-161, change the Programming Note for the **Rect** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinPtInRect — Point In Rectangle

On page 9-169, change the Programming Note for the **Rect** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinQueryQueueStatus — Query Queue Status

On page 9-215, note that this call does not have any possible returns from WinGetLastError.

WinQuerySysValue — Query System Value

On page 9-224, add the following system value identity descriptions to the **Valueid** parameter:

SV_INSERTMODE	(*) TRUE if the system is in insert mode (for edit and multi-line edit controls); FALSE if in overtype mode. This system value is toggled by the system when the insert key is toggled, regardless of which window has the focus at the time.
SV_MENUROLLDOWNDELAY	(*) The delay in milliseconds before displaying a pulldown referred to from a submenu item, when the button is already down as the pointer moves onto the submenu item.
SV_MENUROLLUPDELAY	(*) The delay in milliseconds before hiding a pulldown referred to from a submenu item, when the button is already down as the pointer moves off the submenu item.

WinQueryUpdateRect — Query Update Rectangle

On page 9-230, change the Programming Note for the **Prc** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinQueryWindowRect — Query Window Rectangle

On page 9-242, change the Programming Note for the **Rect** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinQueryWindowUshort — Query Window Short

On page 9-246, change the definitions of the following flags in the **QWS_FLAGS** section of the **b** parameter:

FF_OWNERHIDDEN Frame window is hidden as a result of its owner being hidden or minimized. This indicator is only set if the window and its owner are siblings.

FF_OWNERDISABLED Window's owner is disabled. This indicator is only set if the window and its owner are siblings.

WinReallocMem — Reallocate Memory In Heap

On page 9-248, change the third paragraph of the Remarks section to:

The value of the **MemObj** parameter must have been returned by either the WinAllocMem call or a previous call to WinReallocMem. The memory offset returned is aligned on a *ULONG* boundary.

WinRegisterClass — Register Window Class

On page 9-250, change the first sentence to:

However, if a private class is registered with the same name as one that already exists, the parameters replace the old class parameters and the return value is TRUE.

WinScrollWindow — Scroll Window

On page 9-263, change the Programming Note for the **Scroll** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinSetClassMsgInterest — Set Class Message Interest

On page 9-272, replace the definition of **SMIM_ALL** with:

SMIM_ALL All messages (except for WM_QUIT if **Control** is SMI_AUTODISPATCH or SMI_NOINTEREST).

Add the following to the Remarks section:

The interest for WM_QUIT may not be set to SMI_AUTODISPATCH using SMIM_ALL, because WM_QUIT is the normal means of terminating an application. It may be set specifically if required.

WinSetClipbrdData — Set Clipboard Data

On page 9-273, replace the description of the **FmtInfo** parameter with the following.

FmtInfo *(BIT16)* — input
Information.

Information about the type of data referenced by the **h** parameter.

Memory Model

One and only one of CFI_SELECTOR and CFI_HANDLE must be specified, unless CFI_OWNERDISPLAY is also specified,

CFI_SELECTOR The **h** parameter is a selector in the low order *USHORT*.

Calls return the data object with segment in the low order USHORT, and ZERO in the high order USHORT. This must be manipulated into a far pointer before use, otherwise a protection fault occurs.

When this memory model is specified, the system:

- saves the address (accessing it from the shell process), so that if the setting application terminates normally or abnormally, the data is still available.
- frees the memory from the setting process, so that the setting application may no longer use it.

CFI_SELECTOR must be specified if the **fmt** parameter is CF_TEXT or CF_DSPTEXT.

CFI_HANDLE The **h** parameter is the handle to a metafile or bit map.

This must be specified if the **fmt** parameter is CF_BITMAP, CF_DSPBITMAP, CF_METAFILE or CF_DSPMETAFILE.

Usage Flags

CFI_OWNERFREE Handle is not freed by WinEmptyClipbrd. The application must free the data if necessary.

CFI_OWNERDISPLAY This flag indicates that the format is drawn by the clipboard owner in the clipboard viewer window by means of the WM_PAINTCLIPBOARD message. The **h** parameter should be NULL.

The values may be combined with bit-wise OR. Any number of the usage flags may be specified, but only one of the memory models.

WinSetHook — Set Hook

On page 9-282, add the following to the list of hook chain types :

HK_LOADER See LoaderHook.
HK_MSGCONTROL See MsgCtlHook.

WinSetMsgInterest — Set Message Interest

On page 9-284, replace the definition of **SMIM_ALL** with:

SMIM_ALL All messages (except for WM_QUIT if **Control** is SMI_AUTODISPATCH or SMI_NOINTEREST).

Add the following to the Remarks section:

The interest for WM_QUIT may not be set to SMI_AUTODISPATCH using SMIM_ALL, because WM_QUIT is the normal means of terminating an application. It may be set specifically if required.

WinSetRect — Set Rectangle

On page 9-292, change the Programming Note for the **Rect** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinSetRectEmpty — Set Rectangle Empty

On page 9-293, change the Programming Note for the **Rect** parameter to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinSetSysValue — Set System Value

On page 9-299, add the following system value identity descriptions to the **ValueId** parameter:

SV_INSERTMODE When TRUE, the system is in insert mode.

SV_MENUROLLDOWNDELAY Delay in milliseconds before displaying a pulldown referred to from a submenu item, when the button is already down as the pointer moves onto the submenu item.

SV_MENUROLLUPDELAY Delay in milliseconds before hiding a pulldown referred to from a submenu item, when the button is already down as the pointer moves off the submenu item.

WinSubtractRect — Subtract Rectangle

On page 9-318, change the Notes for the **Dest**, **Src1**, and **Src2** parameters to:

The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinTrackRect – Draw Tracking Rectangle

On page 9-323, replace the Remarks section with:

Remarks

The WinTrackRect call provides general-purpose pointing-device tracking. It draws a rectangle and enables the user to position the entire rectangle, or size a specific side or corner, as required. The resulting rectangle is then returned to the application, which can use this new information for size and position data. The window manager interface for moving and sizing windows by means of the wide sizing borders uses this call for example.

This call enables the caller to control such limiting values as:

- A maximum and minimum tracking size
- Absolute tracking-position limits
- The tracking rectangle side widths
- A restriction of tracking rectangle movements to a predefined positional grid.

It automatically calls WinLockWindowUpdate to prevent output in the window **hwnd** and its descendants while tracking. When tracking has been completed, output is enabled before this call returns. It also determines which button of the pointing device is depressed at the start of the operation, and only completes the tracking operation when the same button is released.

If the **options** parameter of the *TRACKINFO* structure specified by the TF_SETPOINTERPOS value is included, the pointing-device pointer is positioned at the center of the tracking rectangle. Otherwise, the pointing-device pointer is not moved from its current position and a delta is established between the pointing-device position and the part of the tracking rectangle that it moves (the delta is kept constant).

While moving or sizing with the keyboard interface, the pointing-device pointer is repositioned with the tracking rectangle's new size or position.

While tracking, these keys are active:

Enter Accepts the new position or size.

Left cursor Moves the pointing-device pointer and tracking rectangle left.

If the pointing-device pointer is on the upper or lower edge of the tracking rectangle, the pointer is moved to the top-left or bottom-left corner respectively.

Up cursor Moves the pointing-device pointer and tracking rectangle up.

If the pointing-device pointer is on the left or right edge of the tracking rectangle, the pointer is moved to the top-left or top-right corner respectively.

Right cursor Moves the pointing-device pointer and tracking rectangle right.

If the pointing-device pointer is on the upper or lower edge of the tracking rectangle, the pointer is moved to the top-right or bottom-right corner respectively.

Down cursor Moves the pointing-device pointer and tracking rectangle down.

If the pointing-device pointer is on the left or right edge of the tracking rectangle, the pointer is moved to the bottom-left or bottom-right corner respectively.

Esc Cancels the current tracking operation. The value of the tracking rectangle is undefined on exit.

The pointing device and the keyboard interface can be intermixed. The caller need not include the TF_SETPOINTERPOS value to use the keyboard interface, as this value simply initializes the position of the pointing-device pointer.

If TF_GRID is specified in the *TRACKINFO* structure, the interior of the tracking rectangle is restricted to multiples of the values of the **gridwidth** and **gridheight** parameters. The default values for these are the system font character width and half the system font character height respectively.

Tracking movements using the keyboard arrow keys depend on whether or not TF_GRID is specified in the *TRACKINFO* structure. If not specified, the increments are the values of **xkeyboard** and **ykeyboard**. If specified, the increments are the largest multiples of **gridwidth** and **gridheight** that do not exceed **xkeyboard** and **ykeyboard** respectively. If **gridwidth** exceeds **xkeyboard**, or **gridheight** exceeds **ykeyboard**, the keyboard arrow keys do not cause tracking.

The tracking rectangle is usually logically on top of objects it tracks, so that the user can see the old size and position while tracking the new. Thus, it is possible for a window below the tracking rectangle to be updated while part of the tracking rectangle is above it.

Because the tracking rectangle is drawn in exclusive-OR mode, no window can draw below the tracking rectangle (and thereby obliterate it) without first notifying the tracking code, because unwanted areas of the tracking rectangle can be left behind. If the window doing the drawing is clipped out from the window in which the tracking is occurring, this problem does not arise.

To prevent a window that is currently processing a WM_PAINT message drawing over the tracking rectangle, the tracking rectangle is considered a system-wide resource, only one of which can be in use at any time. If there is a risk of the currently-updating window drawing on the tracking rectangle, the tracking rectangle is removed while that window and its child windows update, and it is then replaced. This is done during the WinBeginPaint and WinEndPaint calls. If the tracking rectangle overlaps, it is removed in the WinBeginPaint call. In the WinEndPaint call, all the child windows are updated by means of the WinUpdateWindow call before the tracking rectangle is redrawn.

WinTrackRect has a modal loop within it. The loop has a HK_MSGFILTER hook and a MSGF_TRACK hook code.

Programming Note: It is ensured that the rectangle tracked by this call is within the specified tracking bounds and dimensions. If the rectangle passed is out of these bounds, or it is too large or too small, it is modified to a rectangle that meets these limits.

This call requires the existence of a message queue.

WinUnionRect – Union Rectangle

On page 9-327, change the Notes for the **Dest**, **Src1**, and **Src2** parameters to:

> The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

WinValidateRect – Validate Rectangle

On page 9-331, change the Programming Note for the **Rect** parameter to:

> The value of each field in the RECT structure must be in the range -32 768 through 32 767. The data type WRECT may also be used if it is supported by the language.

Volume 2: (Related Information)

© Copyright IBM Corp. 1990

Chapter 10. Functions Supplied by Applications

Summary of Changes

New	Updated	Section Title
	√	JournalPlaybackHook
	√	MsgFilterHook

JournalPlaybackHook

On page 10-9, change the order of the parameters to:

JournalPlaybackHook (hab, Skip, qmsg, *Time*)

MsgFilterHook

On page 10-15, add the following item to the **Context** parameter:

MSGF_DRAG Direct manipulation mode loop.

© Copyright IBM Corp. 1990

Chapter 12. Default Window Procedure Message Processing

Summary of Changes

New	Updated	Section Title
	√	WM_HITTEST
	√	WM_SETSELECTION

WM_HITTEST

On page 12-23, change the **HT_ERROR** description in the **result** parameter to:

HT_ERROR As HT_DISCARD, except that if the message is a button-down message, an alarm sounds and the window concerned is brought to the foreground.

WM_SETSELECTION

On page 12-40, add the following to the Remarks section:

This message is sent to a window when it loses the focus to another window that it does not own. It allows an application to remove the selection when the focus is removed to another application, but to keep it if, for example, the same application displays a dialog box.

© Copyright IBM Corp. 1990

Chapter 13. Button Control Window Processing

Summary of Changes

New	Updated	Section Title
	√	Button Control Styles

Button Control Styles

On page 13-1, add the following to the list of button control styles:

This style can be ORed with the BS_AUTORADIOBUTTON style:

BS_NOCURSORSELECT The radio button does not select itself when given the focus as the result of an arrow key or tab key.

Also add the following to the description of the **BS_NOPOINTERFOCUS** style:

This style has no effect on keyboard interaction. The tab key can still be used as usual to move the focus to the button.

© Copyright IBM Corp. 1990

Chapter 14. Entry Field Control Window Processing

Summary of Changes

New	Updated	Section Title
	√	Entry Field Control Styles

Entry Field Control Styles

On page 14-1, add the following style descriptions to the list of entry field control styles:

ES_AUTOSCROLL If the user tries to move off the end of a line, the control automatically scrolls one-third the width of the window in the appropriate direction.

ES_COMMAND This style identifies the entry field as a command entry field. This information is used by the Help Manager to provide command help if the end user requests help for this field.

Not more than one entry field on each dialog should be given this style.

ES_AUTOTAB This style indicates that when the field is filled by adding a character to the end of the entry-field text, the effect of a tab key will be generated. Inserting or replacing a character in the middle of the text, however, does not result in an autotab.

This style is recommended for use with fixed-length, non-scrollable fields that are filled completely. The maximum length of the entry-field text is held in the control data (see Entry Field Control Data).

Change the definition of ES_UNREADABLE to:

This style causes the text to be displayed as an asterisk for each character. It can be used for passwords.

© Copyright IBM Corp. 1990

Chapter 15. Frame Control Window Processing

Summary of Changes

New	Updated	Section Title
	√	WM_FORMATFRAME

WM_FORMATFRAME

On page 15-11, add the following sentence to the Default Processing section:

If not processed by the client, the frame control window procedure calculates the size and position of all of the standard frame controls.

© Copyright IBM Corp. 1990

Chapter 16. List Box Control Window Processing

Summary of Changes

New	Updated	Section Title
	√	List Box Control Styles
	√	WM_DRAWITEM
	√	WM_MEASUREITEM
	√	LM_QUERYCURSOR
	√	LM_SETCURSOR
	√	LM_SETSELECTION

List Box Control Styles

On page 16-1, delete LS_NOVERTSCROLL from the list of list-box styles, as it is not defined.

WM_DRAWITEM

On page 16-3, add the following to the Remarks section:

The item text must not be changed during the processing of this message.

WM_MEASUREITEM

On page 16-4, change the Remarks section to:

This message is sent to the owner of a list box that has a style of LS_OWNERDRAW, to offer the owner an opportunity to establish the height and width (for a horizontally scrollable list box control) of an item that accommodates any special requirements for the drawing of items in that list box. It is sent when items in the list box are inserted or deleted, and also when presentation parameters for the list box change.

All items in a list box must have the same height, which must be greater than or equal to the height of the current font.

LM_QUERYCURSOR

On page 16-7, delete this message.

LM_SETCURSOR

On pages 16-12 and 16-13, delete this message.

© Copyright IBM Corp. 1990

LM_SETSELECTION

On page 16-15, delete this message.

Chapter 17. Menu Control Window Processing

Summary of Changes

New	Updated	Section Title
	√	MM_DISMISSMENU
	√	MM_QUERYITEMTEXT
	√	MM_SELECTITEM

MM_DISMISSMENU

On page 17-8, delete this message.

MM_QUERYITEMTEXT

On page 17-14, change the **maxcount** parameter to:

maxcount *(SHORT)*
Maximum count:

Copy the item text as a null-terminated string, but limit the number of characters copied, including the null termination character, to this value, which must be greater than zero.

MM_SELECTITEM

On page 17-17, change the **param2** parameter to the following.

param2

reserved *(BIT16)*
Reserved.

NULL Reserved value.

dismissed *(BOOL)*
Dismissed flag:

TRUE Dismiss the menu
FALSE Do not dismiss the menu.

© Copyright IBM Corp. 1990

Chapter 18. Multi-Line Entry Field Control Window Processing

Summary of Changes

New	Updated	Section Title
	√	WM_CONTROL
	√	MLM_FORMAT
	√	MLM_QUERYSEL
	√	MLM_SEARCH
	√	MLM_SETFORMATRECT

WM_CONTROL

On page 18-6, change the following values of the **errinfo** parameter:

EFR_RESIZE should be MLFEFR_RESIZE
EFR_TABSTOP should be MLFEFR_TABSTOP
EFR_FONT should be MLFEFR_FONT
EFR_WORDWRAP should be MLFEFR_WORDWRAP
EFR_TEXT should be MLFEFR_TEXT
ETL_TEXTBYTES should be MLFETL_TEXTBYTES

On page 18-7, change the following values of the **errind** parameter:

EFR_TOOMUCHTEXT should be MLFCPBD_TOOMUCHTEXT
EFR_CLPBDERROR should be MLFCPBD_CLPBDERROR

These values should appear before the line that reads:

For a **notifycode** of MLN_MARGIN

MLM_FORMAT

On page 18-12, change the following values of the **format** parameter:

MLE_CFTEXT should be MLFIE_CFTEXT
MLE_NOTRANS should be MLFIE_NOTRANS
MLE_WINFMT should be MLFIE_WINFMT

All MLE_* constants referred to in this chapter should be MLFIE_* constants.

MLM_QUERYSEL

On page 18-21, change the following values of the **querymode** parameter:

0 should be MLFQS_MINMAXSEL
1 should be MLFQS_MINSEL
2 should be MLFQS_MAXSEL
3 should be MLFQS_ANCHORSEL
4 should be MLFQS_CURSORSEL

These changes also apply to the values given under the **reply** parameter.

© Copyright IBM Corp. 1990

MLM_SEARCH

On page 18-26, change the following values of the **style** parameter:

MLSS_CASESENSITIVE should be MLFSEARCH_CASESENSITIVE
MLSS_SELECTMATCH should be MLFSEARCH_SELECTMATCH
MLSS_CHANGEALL should be MLFSEARCH_CHANGEALL

MLM_SETFORMATRECT

On page 18-30, change the following values of the **flags** parameter:

MLESFR_MATCHWINDOW should be MLFFMTRECT_MATCHWINDOW
MLESFR_LIMITHORZ should be MLFFMTRECT_LIMITHORZ
MLESFR_LIMITVERT should be MLFFMTRECT_LIMITVERT

Chapter 19. Prompted Entry Field Control Window Processing

Summary of Changes

New	Updated	Section Title
	√	WM_CONTROL
	√	Combo Box Control Window Messages
	√	WM_UPDATESTYLE

WM_CONTROL

On page 19-2, change the description of CBN_ENTER in the **notifycode** parameter to:

> The user has pressed the ENTER key or double clicked (single clicked in the case of a drop-down list) on an item in the list box control.

Combo Box Control Window Messages

On page 19-3, delete references to the LM_QUERYCURSOR, LM_SETCURSOR, and LM_SETSELECTION messages. These messages are not defined.

WM_UPDATESTYLE

On page 19-6, delete this message.

© Copyright IBM Corp. 1990

Chapter 20. Scroll Bar Control Window Processing

Summary of Changes

New	Updated	Section Title
	√	SBM_QUERYHILITE
	√	SBM_SETHILITE
	√	SBM_SETSCROLLBAR

SBM_QUERYHILITE

On page 20-4, delete this message.

SBM_SETHILITE

On page 20-5, delete this message.

SBM_SETSCROLLBAR

On page 20-7, add the following to the Remarks section:

For example, if a scroll-bar is to allow scrolling through 100 lines of text, of which 50 are visible at any one time, and the top display line is currently the 25th, **first** should be set to 1, **last** to 51 (since there are only 51 positions at which the slider may be placed), and **slider** to 25. The SBM_SETTHUMBSIZE message should be used in this example to set the slider size to 50 visible parts out of 100.

© Copyright IBM Corp. 1990

Chapter 20.1. Spin Button Control Window Processing

Summary of Changes

New	Updated	Section Title
√		Types of Spin Button Control
√		SPBM_OVERRIDESETLIMITS
√		SPBM_QUERYLIMITS
√		SPBM_QUERYVALUE
√		SPBM_SETARRAY
√		SPBM_SETCURRENTVALUE
√		SPBM_SETLIMITS
√		SPBM_SETMASTER
√		SPBM_SETTEXTLIMIT
√		SPBM_SPINDOWN
√		SPBM_SPINUP
√		WM_CONTROL (in Spin Controls)

This system-provided window procedure processes the actions of a spin button control (WC_SPINBUTTON).

Types of Spin Button Control

A spin button can be one of the following:

SPBS_MASTER The spin button component consists of at least one single line entry field (SLE), or spin field, and two arrows, the Up Arrow and the Down Arrow. When a spin button contains more than one spin field, the master component contains the spin arrows. If the component contains only one spin field, it should be a master.

SPBS_SERVANT You can create a multi-field spin button by spinning servants from the master.

Create a spin button using the style bits listed below. These styles can be ORed together.

Specify the type of characters allowed in the spin field:

SPBS_ALLCHARACTERS Any character can be typed in the spin field. This is the default.

SPBS_NUMERICONLY Only the digits 0—9 and – can be typed in the spin field.

SPBS_READONLY Nothing can be typed in the spin field. The selection cursor changes from a vertical bar to a dotted box around the text.

Specify how the text is to be presented in the spin field:

© Copyright IBM Corp. 1990

SPBS_JUSTLEFT Left justify. This is the default.

SPBS_JUSTRIGHT Right justify.

SPBS_JUSTCENTER Centered text.

When you do not want a border around the component, specify:

SPBS_NOBORDER Suppresses drawing a border.

To increase the spin speed, specify:

SPBS_FASTSPIN Enables the spin button to increase the spin speed with time. The speed doubles every two seconds.

Programming Note: The spin button skips information when this option is specified. Do not use SPBS_FASTSPIN if the application requires that this field be checked each time a spin up or spin down occurs. Do not specify this option on a master component that has servants spun from it.

You can pad numeric fields with zeros. This is useful when the spin field contains values that represent time or money. Specify:

SPBS_PADWITHZEROS The output number is padded at the front between the first non-zero digit and the field width, or 11 characters, whichever is the lesser. The negative sign, if there is one, is retained. The maximum number of characters required to display a *LONG* number is 11.

SPBM_OVERRIDESETLIMITS

This message causes the component to set or reset numeric limits.

Parameters

param1

uplimit *(LONG)*
Upper limit.

param2

lowlimit *(LONG)*
Lower limit.

Returns

reply

result *(BOOL)*
Return:

TRUE Successful completion.
FALSE Error occurred.

Remarks

The application sends this message to the component to set or reset numeric limits.

This message is functionally identical to SPBM_SETLIMITS, except that the current value of the spin button does not change if it is out of range.

When the upper limit is less than the lower limit, FALSE is returned.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

SPBM_QUERYLIMITS

This message enables an application to query the limits of a numeric spin field.

Parameters

param1

uplimit *(LONG)*
Upper limit.

param2

lowlimit *(LONG)*
Lower limit.

Returns

reply

result *(BOOL)*
Return:

TRUE	Successful completion.
FALSE	Error occurred.

Remarks

The application sends this message to the component to determine the limits of a numeric spin field.

When the spin button has no data, or when it is spinning an array, FALSE is returned.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

SPBM_QUERYVALUE

This message causes the component to show the value in the spin field.

Parameters

param1

storage *(STORAGE)*
A place for the returned value.

This value is either the address of a string or the address of a long variable. If the **bufsize** is NULL, **param1** is assumed to be an address of a long variable. If **param1** is Other, it is assumed to be an address of a string.

NULL Causes the spin button to process the reset or update as specified, but it will not try to return a value to the application.
Other The address where the value is returned.

param2

Consists of two USHORT parameters.

bufsize *(USHORT)*
Buffer size.

If **bufsize** is too small to return all of the text, the spin button returns as much of the text as it can.

NULL The spin button assumes that **param1** is the address of a long variable. If the data in the spin button is spinning between an upper and lower limit, the current value is passed back in the variable.

If the data in the spin button is in an array, the index of the current array value (or last valid value) is passed back in the variable.
Other The spin button assumes that **param1** is the address of a string. The information passed back in the string is dependent upon the flags in the **value** field.

value *(USHORT)*
Update/reset value

Controls how the spin field is updated.

SPBQ_UPDATEIFVALID Update the contents of the spin field if the value is valid. This is the default.

Specifying this flag on a query will *not* update the contents of the spin field if it is *exactly* the same as an item in the spin button list.

If an item in the list is Monday, specifying SPBQ_UPDATEIFVALID updates the spin field contents when MONDAY, monday, or mONDAY are typed, but not when Monday is typed. This prevents recursion if the application checks for the validity each time a SPBN_CHANGE message is sent from the component.

SPBQ_ALWAYSUPDATE Update the contents of the spin field if the value is valid. Reset the contents of the spin field to the last valid value if the field contains data that is not valid.

If the spin button is spinning numbers between an upper and a lower limit, and the content of the spin field is a valid number that is out of range, the spin button does not reset itself to the last valid value. It sets the current position at the upper limit when the out-of-range number specified is above the upper limit. It sets the current position at the lower limit when the out-of-range number is below the lower limit.

When the current value is changed, the return of the query message is still FALSE.

SPBQ_DONOTUPDATE Do not update the contents of the spin field, even if the value is valid.

Returns

reply

result *(BOOL)*
Return:

TRUE Successful completion.
FALSE Error occurred.

Remarks

The application sends this message to the component to determine what value is in the spin field. The application sets up a field for the component to deposit the value, and sets a flag to determine what the function does when the value matches or does not match the given spin-list values.

TRUE is returned when a matched value is found, or the data is in the range.

FALSE is returned when no match is found, the value is out of range, or no spin data exists.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

SPBM_SETARRAY

This message causes the component to set or reset the array of data.

Parameters

param1

strl *(STRL)*
The new array of values.

param2

items *(COUNT2)*
Number of items in the array.

Returns

reply

result *(BOOL)*
Return:

TRUE	Successful completion.
FALSE	Error occurred.

Remarks

The application sends this message to the component to set or reset the array of data.

The component tries to leave the current value unchanged. However, if the current value is out of range for the new array, it is moved to the closest extreme. Thus, if the current value is less than 0, it is moved to 0. If the current value is greater than the previous value, it is set to the previous value.

If the data exceeds 64KB, or if **param1** or **param2** equal zero, FALSE is returned.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

SPBM_SETCURRENTVALUE

This message causes the component to set or reset the current numeric value or array index.

Parameters

param1

value *(LONG)*
Current value or index of array.

param2

reserved *(BIT32)*
Reserved.

NULL Reserved value.

Returns

reply

result *(BOOL)*
Return:

TRUE Successful completion.
FALSE Error occurred.

Remarks

The application sends this message to the component to set or reset the current numeric value or array index.

FALSE is returned when the value is out of range or there is no spin data.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

SPBM_SETLIMITS

This message causes the component to set or reset numeric limits.

Parameters

param1

uplimit *(LONG)*
Upper limit.

param2

lowlimit *(LONG)*
Lower limit.

Returns

reply

result *(BOOL)*
Return:

TRUE Successful completion.
FALSE Error occurred.

Remarks

The application sends this message to the component to set or reset numeric limits. The component sets the current value to the content in the spin field when it is a valid number. When the current value is out of the range of the limits, it is moved to the nearest limit, upper or lower.

If the upper limit is less than the lower limit, FALSE is returned.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

SPBM_SETMASTER

This message causes the component to identify its master.

Parameters

param1

hwnd *(HWND)*
Handle of master component.

param2

reserved *(BIT32)*
Reserved.

NULL Reserved value.

Returns

reply

result *(BOOL)*
Return:

TRUE Successful completion.
FALSE Error occurred.

Remarks

The application sends this message to the component to identify the component's master.

Programming Note: When the application wants to take control of the spin button, it must set the **param1** of each spin button to NULL. This must be done when, for example, a spin button with a non-contiguous list of spin values is created (2, 4, 6, 8, 10...). When the **param1** of a spin button is NULL, the spin button does not perform the following default functions:

- Spin up or down on its own when the Up or Down Arrow key is pressed.
- Spin up or down when the Up or Down Arrow of the master is pressed.
- A master does not take the focus when its arrows are pressed and none of its servants have focus.

- The spin button does not send itself an SPBM_QUERYVALUE message with the SPBQ_ALWAYSUPDATE flag to update the current value when an SPBM_SPINUP or SPBM_SPINDOWN message is received.
- The spin button does not fast spin.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

SPBM_SETTEXTLIMIT

This message sets the maximum number of characters allowed in a spin field.

Parameters

param1

llmlt *(USHORT)*
Number of characters to allow.

param2

reserved *(BIT32)*
Reserved.

NULL Reserved value.

Returns

reply

result *(BOOL)*
Return:

TRUE Successful completion.
FALSE Error occurred.

Remarks

The application sends this message to set the maximum number of characters allowed in the spin field. The size limit of the spin field is 255 characters. This is the default.

When the size exceeds 255 characters, FALSE is returned,

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

SPBM_SPINDOWN

This message causes the component to show the previous value (spin backward).

Parameters

param1

item *(ULONG)*
Number of values to spin down.

param2

reserved *(BIT32)*
Reserved.

NULL Reserved value.

Returns

reply

result *(BOOL)*
Return:

TRUE Successful completion.
FALSE Error occurred.

Remarks

The application sends this message to the component when it wants the previous value shown (spin backward).

When there is no data to spin, FALSE is returned.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

SPBM_SPINUP

This message causes the component to show the next value (spin forward).

Parameters

param1

item *(ULONG)*
Number of values to spin up.

param2

reserved *(BIT32)*
Reserved.

NULL Reserved value.

Returns

reply

result *(BOOL)*
Return:

TRUE Successful completion.
FALSE Error occurred.

Remarks

The application sends this message to the component when it wants the next value shown (spin forward).

When there is no data to spin, FALSE is returned.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

WM_CONTROL (in Spin Controls)

For the cause of this message, see WM_CONTROL in the *Presentation Manager Programming Reference: Volume 2*.

Parameters

param1

id *(USHORT)*
Identity of the spin button component window.

notifycode *(USHORT)*
Notification code:

SPBN_UPARROW Tells the application that the Up Arrow was clicked on, or the Up Arrow key was pressed.
SPBN_DOWNARROW Tells the application that the Down Arrow was clicked on, or the Down Arrow key was pressed.
SPBN_SETFOCUS Tells the application which spin field was selected.
SPBN_KILLFOCUS Tells the application when the spin field loses focus.
SPBN_ENDSPIN Tells the application that the user released mouse button 1 or one of the arrow keys while spinning a button.
SPBN_CHANGE Tells the application that the contents of the spin field changed.

param2

hwnd *(HWND)*
Window handle.

The interpretation of this handle is dependent upon the following notification codes:

- SPBN_UPARROW, SPBN_DOWNARROW, and SPBN_ENDSPIN.

 param2 is the handle to the currently selected spin field in a particular master-servant setup. If either the Up or Down Arrow is clicked on and none of a spin button's servants are currently selected, the master will return a handle to itself.

- SPBN_SETFOCUS

 param2 is the handle of the currently selected spin field.

 This message tells the application which spin field is selected.

- SPBN_KILLFOCUS

 param2 is NULL if the spin field loses focus or no spin field is currently selected.

 This message tells the application when a spin field loses focus.

 Programming Note: Both SPBN_KILLFOCUS and SPBN_SETFOCUS are set independently. You must check this message only when the application does not specify a master-servant relationship.

- SPBN_CHANGE

 param2 is the handle of the spin button in which the spin field text changed.

Returns

reply *(BIT32)*
Reserved.

NULL Reserved value.

Remarks

This message is sent when, as specified by **notifycode**, the spin button component must tell its owner of a significant event.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

Chapter 23.1. Direct Manipulation Messages

Summary of Changes

New	Updated	Section Title
√		DM_DRAGERROR
√		DM_DRAGFILECOMPLETE
√		DM_DRAGLEAVE
√		DM_DRAGOVER
√		DM_DROP
√		DM_DROPHELP
√		DM_EMPHASIZETARGET
√		DM_ENDCONVERSATION
√		DM_FILERENDERED
√		DM_PRINT
√		DM_RENDER
√		DM_RENDERCOMPLETE
√		DM_RENDERFILE
√		DM_RENDERPREPARE

DM_DRAGERROR

This message is sent to the caller of DrgDragFiles or DrgAcceptDroppedFiles when an error occurs during a move or copy operation for a file.

Parameters

param1

error *(USHORT)*
Error Code.

Returned from DosCopy, DosMove, or DosDelete.

operation *(USHORT)*
Operation that failed:

DFF_MOVE	DosMove failed.
DFF_COPY	DosCopy failed.
DFF_DELETE	DosDelete failed.

param2 *(HSTR)*
HSTR of file contributing to the error.

© Copyright IBM Corp. 1990

Returns

reply *(HSTR)*
Action indicator:

DME_IGNORECONTINUE	Do not retry the operation, but continue with the rest of the files.
DME_IGNOREABORT	Do not retry the operation, and do not try any other files.
DME_RETRY	Retry the operation.
DME_REPLACE	Replace the file at the destination. Used if FALSE is not specified.
Other	*HSTR* of new file name to use for retry.

Remarks

The receiver of this message should return the action that the sender should take.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

DM_DRAGFILECOMPLETE

This message is sent when a direct manipulation operation on a file or files is complete.

Parameters

param1 *(HSTR)*
File handle.

param2 *(USHORT)*
Flags set as follows:

DF_MOVE	The operation was a move. If this flag is not set, the operation was a copy.
DF_SOURCE	The receiving window was the source of the drag. If this flag is not set, the receiver was the target of the drop.
DF_SUCCESSFUL	The drag operation was successful for the file. If this flag is not set, the operation failed.

Returns

reply *(BIT32)*
Reserved.

NULL Reserved value.

Remarks

param1 is *HSTR* for the source file if this message is sent by DrgDragFiles, and is *HSTR* for the target file if this message is sent by DrgAcceptDroppedFiles.

This message is sent by DrgDragFiles to its caller when the move or copy operation is completed, regardless of success or failure. It is also sent by DrgAcceptDroppedFiles when a file has been successfully dropped on the caller.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

DM_DRAGLEAVE

This message is sent to a window that is being dragged over when one of these conditions occur:

1. The object is dragged outside the boundaries of the window.
2. The drag operation is terminated while the object is over the window.

Parameters

param1

dragInfo *(DRAGINFO)*
DRAGINFO structure for the drag operation.

param2 *(BIT32)*
Reserved.

NULL Reserved value.

Returns

reply *(BIT32)*
Reserved.

NULL Reserved value.

Remarks

This message allows for target emphasis and de-emphasis during the direct manipulation process. This message is not sent when a drop occurs. Use DM_DROP as a signal to remove the target emphasis.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action on it other than to return FALSE.

DM_DRAGOVER

This message allows the window under the mouse pointer to determine if the object or objects currently being dragged can be dropped.

Parameters

param1

dragInfo *(DRAGINFO)*
DRAGINFO structure representing the object being dragged.

param2
Mouse pointer location.

xdrop *(SHORT)*
x-coordinate of the mouse pointer in desktop coordinates.

ydrop *(SHORT)*
y-coordinate of the mouse pointer in desktop coordinates.

Returns

reply

drop *(USHORT)*
Drop indicator.

DOR_DROP	Object can be dropped. When this reply is given, **defaultop** must be set to indicate which operation will be performed if the user should drop at this location. This is used to provide visual feedback to the user.
DOR_NODROP	Object cannot be dropped at this time. The target can accept the object in the specified type and format using the specified operation, but the current state of the target will not allow it to be dropped on. The target may change state in the future so that the same object may be acceptable.
DOR_NODROPOP	Object cannot be dropped at this time. The target can accept the object in the specified type and format, but the current operation is not acceptable. A change in the drag operation may change the acceptability of the object.
DOR_NEVERDROP	Object cannot be dropped. The target cannot accept the object now and will not change state so that the object will be acceptable in the future. If this response is returned, no more DM_DRAGOVER messages will be sent to the target until the pointer is moved out of and back into the target window.

defaultop *(USHORT)*
Target-defined default operation.

DO_COPY	Operation is a copy.
DO_MOVE	Operation is a move.
DO_UNKNOWN	Operation is application-defined.

Remarks

This message is sent to the window that is directly under the hot spot of the mouse pointer during the drag operation when any of the following conditions are met:

1. The user moves the mouse.
2. A key is pressed.
3. A WM_BUTTON1UP, WM_BUTTON2UP, or WM_BUTTON3UP message is received, indicating that the direct manipulation operation corresponds to the **VkTerminate** parameter specified by the source on the call to DrgDrag. In this case the message is sent only if the mouse has moved since the last DM_DRAGOVER message was sent.

The receiver can gain access to **draginfo** with DrgAccessDraginfo. The acceptability of the dragged objects can be determined by querying the **type** and **rmf** string handles in each of the *DRAGITEM* structures carried in **draginfo**.

The receiver should provide target emphasis for itself if it returns anything other than DOR_NEVERDROP for this message. The receiver can use DrgSetDragPointer to change the bit map while it is being dragged over.

If **operation** in *DRAGINFO* is DO_DEFAULT and the target returns DOR_DROP for **drop**, **defaultop** should be set to reflect what the target defines as the default operation. This information is used to provide the appropriate modification to the drag pointer.

If **defaultop** is not one of the defined values, **drop** is treated as DOR_NEVERDROP.

If **operation** in *DRAGINFO* is not DO_DEFAULT or **drop** is not DOR_DROP, then **defaultop** is ignored.

Default Processing

The WinDefWindowProc returns DOR_NEVERDROP to the sender of this message.

DM_DROP

This message is sent to the target when the dragged object is dropped.

Parameters

param1

draginfo *(DRAGINFO)*
DRAGINFO structure.

param2 *(BIT32)*
Reserved.

NULL Reserved value.

Returns

reply *(BIT32)*
Reserved.

NULL Reserved value.

Remarks

This message is sent to the window directly under the hot spot of the mouse pointer at the completion of a direct manipulation operation only if DOR_DROP was returned for the DM_DRAGOVER message sent to the window during the drag.

The receiver can obtain access to **draginfo** with DrgAccessDraginfo.

The receiver should immediately remove any target emphasis and post a private message to itself to initiate the data transfer conversations needed to complete the operation.

The receiver should use the **xoffset**, and **yoffset**, fields in the *DRAGITEM* structure to position the dropped object within its window relative to the drop point, so that no movement of the dragged image is perceived by the user when the drop occurs.

When the application receiving the DM_DROP message has finished all data transfer operations, it should free the *DRAGINFO* structure using DrgFreeDraginfo.

Default Processing

The WinDefWindowProc calls DrgDeleteDraginfoStrHandles and DrgFreeDraginfo for **draginfo** and will return FALSE.

DM_DROPHELP

This message requests help for the current drag operation.

Parameters

param1

draginfo *(DRAGINFO)*
DRAGINFO structure used in the drag operation.

param2 *(BIT32)*
Reserved.

NULL Reserved value.

Returns

reply *(BIT32)*
Reserved.

NULL Reserved value.

Remarks

This message is posted to the target of a drop when F1 is pressed during a direct manipulation operation.

The **operation** member of **draginfo** can be used to provide help information in the context of the drag operation during which it was requested.

Default Processing

The WinDefWindowProc calls DrgDeleteDraginfoStrHandles and DrgFreeDraginfo for **draginfo** and returns false.

DM_EMPHASIZETARGET

This message is sent to the caller of DrgAcceptDroppedFiles to inform it to either apply or remove target emphasis from itself.

Parameters

param1

x *(SHORT)*
x-coordinate of mouse pointer, in window coordinates.

y *(SHORT)*
y-coordinate of mouse pointer, in window coordinates.

param2 *(BOOL)*
Flags:

TRUE Apply emphasis.
FALSE Remove emphasis.

Returns

reply *(BIT32)*
Reserved.

NULL Reserved value.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

DM_ENDCONVERSATION

The target uses this message to notify a source that a drag operation is complete.

Parameters

param1

itemid *(ULONG)*
Item ID.

The **itemid** from the *DRAGITEM* that was contained within the *DRAGINFO* structure when the object was dropped.

param2 *(ULONG)*
Flags set as follows:

DMFL_TARGETSUCCESSFUL The target successfully completed its portion of the rendering operation.
DMFL_TARGETFAIL The target failed to complete its portion of the rendering operation.

Returns

reply *(BIT32)*
Reserved.

NULL Reserved value.

Remarks

This message is used to inform a source that the target has completed its part of a rendering operation. It is sent by the target to the source.

The target must send this message under any of the following circumstances:

- The target receives a DM_RENDERCOMPLETE message and will not retry the operation.
- The target completes the rendering operation without involvement from the source.
- The target wants to terminate a rendering operation in progress.
- The target chooses not to render an object that was dropped on it.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

DM_FILERENDERED

This message is sent to the window handling the drag conversation for the caller of DrgDragFiles.

Parameters

param1 *(RENDERFILE)*
Pointer to a *RENDERFILE* structure.

param2 *(BOOL)*
Flags:

TRUE Operation succeeded.
FALSE Operation failed.

Returns

reply *(BIT32)*
Reserved.

NULL Reserved value.

Remarks

This message is sent when the rendering (moving or copying) of a file is complete. This window's handle is the **dragfiles** field of the *RENDERFILE* structure sent on DM_RENDERFILE.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

DM_PRINT

This message is sent to a source to request it to print the current view of an object.

Parameters

param1

dItem *(DRAGITEM)*
DRAGITEM structure.

param2

queuename *(HSTR)*
String handle representing the name of the printer queue the object is sent to.

Returns

reply *(BIT32)*
Reserved.

NULL Reserved value.

Remarks

The target sends this message to a source window to request that an object be printed. The printer should not allow a drop unless DRM_PRINT is a valid rendering mechanism for the object.

The sender is responsible for creating **queuename** using DrgAddStrHandle. The sender must issue DrgDeleteStrHandle for the string handle when it regains control after sending the message to the source. The source should make a copy of this string for its own use, if necessary, before replying to this message.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

DM_RENDER

This message is used to request a source to provide a rendering of an object in a specified rendering mechanism and format.

Parameters

param1

dxfer *(DRAGTRANSFER)*
DRAGTRANSFER structure.

param2 *(BIT32)*
Reserved.

NULL Reserved value.

Returns

success *(BOOL)*
Success indicator:

TRUE	Successful completion.
FALSE	Error occurred.

Remarks

The target sends this message to a source window to request a rendering of an object. If the source returns FALSE, it may set flags in the *DRAGTRANSFER* structure that tell the target how to perform the rendering operation on its own, or how to retry the operation. If no flags are set, the source will not allow a rendering of the object.

If TRUE is returned, the message was processed by the recipient and the requested rendering will take place. The source will post a DM_RENDERCOMPLETE message to the target when the rendering is complete.

If FALSE is returned, either the message was not processed by the recipient, or the recipient could not perform the requested rendering. See **reply** in *DRAGTRANSFER* for more information.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

DM_RENDERCOMPLETE

This message is posted by a source to a target window. It informs the target that the source has completed a requested rendering operation.

Parameters

param1

dxfer *(DRAGTRANSFER)*
DRAGTRANSFER structure. This must be the same pointer that was sent to the source in the DM_RENDER message.

param2

fs *(USHORT)*
Flag field indicating successful completion set as follows:

DMFL_RENDERFAIL	The source is unable to perform the rendering operation. The target may be allowed to retry. If the target is allowed to retry and chooses not to, it must send a DM_ENDCONVERSATION message to the source.
DMFL_RENDEROK	The source has completed the rendering operation. When the target completes its part of the rendering operation, it must post a DM_RENDERCOMPLETE message to the source.

DMFL_RENDERRETRY The source has completed the rendering operation and will allow the target to retry its part of the operation if it fails. This flag can be set in conjunction with either the DMFL_RENDERFAIL or DMFL_RENDEROK flags.

Returns

reply *(BIT32)*
Reserved.

NULL Reserved value.

Remarks

If the rendering operation failed for an intermittent reason, the source can allow the target to retry the operation. The source should return to the state it was in when the drop occurred for that object. The target resumes the rendering operation from the beginning.

If the rendering operation encounters a permanent failure, the source should fail the operation and proceed as if the rendering was completed.

If the rendering operation completes successfully, the source should return to the state it was in when the drop occurred for that object. This allows the target to retry the operation if its portion of the rendering failed. The target must post a DM_ENDCONVERSATION message when:

- It determines that the rendering operation successfully completed, or
- It chooses not to retry a rendering operation that failed.

Default Processing

The WinDefWindowProc should send a DM_ENDCONVERSATION message to the window indicated in the **item** field of the *DRAGITEM* structure. The message should indicate that the target failed in its part of the rendering operation. Sending the DM_ENDCONVERSATION message allows the source to release the resources it dedicated to the rendering operation.

DM_RENDERFILE

This message is sent to the caller of DrgDragFiles to tell it to render a file.

Parameters

param1 *(RENDERFILE)*
Pointer to a *RENDERFILE* structure.

param2 *(BIT32)*
Reserved.

NULL Reserved value.

Returns

reply *(BOOL)*
Render handling:

TRUE The receiver handled the rendering.
FALSE DrgDragFiles should render this file.

Remarks

This message is sent when TRUE is specified in DrgDragFiles. The receiver should perform the operation indicated by the TRUE field in the *RENDERFILE* structure, moving or copying **source** to **target**.

When the operation is complete, a DM_FILERENDERED message should be sent to **dragfiles** window.

The *RENDERFILE* structure is allocated temporarily for the receiver of this message. The receiver should make a copy if it needs to use the data in this structure after returning.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

DM_RENDERPREPARE

This message tells a source to prepare for the rendering of an object.

Parameters

param1

dxfer *(DRAGTRANSFER)*
DRAGTRANSFER structure.

param2 *(BIT32)*
Reserved.

NULL Reserved value.

Returns

success *(BOOL)*
Success indicator:

TRUE The message was processed by the recipient and it is ready to perform the rendering operation. The target of the drop sends a DM_RENDER message to request the rendering with a specific rendering mechanism and format.
FALSE The message either was not processed by the recipient, or it is unprepared to perform the rendering. The **item** field in *DRAGITEM* may not be properly initialized, and therefore the target should not send a DM_ENDCONVERSATION message.

Remarks

This message must be sent when DC_PREPARE is on in the *DRAGITEM* structure.

This message is used to allow the source to create an invisible window to handle the conversation required for the data transfer.

Default Processing

The WinDefWindowProc does not expect to receive this message and takes no action other than to return FALSE.

Chapter 24. Dynamic Data Exchange Messages

Summary of Changes

New	Updated	Section Title
	√	WM_DDE_INITIATE
	√	WM_DDE_INITIATEACK

WM_DDE_INITIATE

On page 24-4, add the following to the Remarks section:

A modal window, such as a message box, must not be invoked during the processing of this message.

WM_DDE_INITIATEACK

On page 24-5, add the following:

Remarks

A modal window, such as a message box, must not be posted during the processing of this message.

© Copyright IBM Corp. 1990

Chapter 25. Help Manager Messages

Summary of Changes

New	Updated	Section Title
	√	HM_ACTION_BAR_COMMAND

HM_ACTION_BAR_COMMAND

On page 25-1, this message is now named HM_ACTIONBAR_COMMAND.

© Copyright IBM Corp. 1990

Chapter 26. Resource Files

Summary of Changes

New	Updated	Section Title
	√	Directives
	√	Menu Template
	√	Predefined Control Statements

Directives

On page 26-4, in the Directives section replace all occurrences of "#rcinclude" with "rcinclude."

Menu Template

On page 26-14, in the **Version** parameter note that versions 0 and 1 are valid.

Replace the **Reserved** parameter description with the following:

Presentation parameters offset *(USHORT)*
Offset of presentation parameters from the start of the template, in bytes. This field does not exist for version 0 of the template.

Predefined Control Statements

On page 26-25, replace the example of a WINDOWTEMPLATE statement with the following. Changes are indicated by a vertical bar.

```
WINDOWTEMPLATE wind1
BEGIN
 FRAME "My Window", 1, 10, 10, 320, 130, WS_VISIBLE,
       FCF_STANDARD | FCF_VERTSCROLL
 BEGIN
  WINDOW "", FID_CLIENT, 0, 0, 0, 0, "MyClientClass", style
 END
END
```

© Copyright IBM Corp. 1990

Chapter 28. Code Pages

Summary of Changes

New	Updated	Section Title
	√	Full Screen VIO Applications
	√	Font Layout

Full Screen VIO Applications

On page 28-1, add code page 1004 (DeskTop Publishing) to the list of supported ASCII code pages. The contents of the code page are shown in Figure 1.

	Char Set	Code Page
Canadian-French	993	863
DeskTop Publishing	1146	1004
Multilingual	980	850
Nordic	995	865
Portuguese	990	860
U.S. (IBM PC)	919	437

1 → 2 ↓	0-	1-	2-	3-	4-	5-	6-	7-		8-	9-	A-	B-	C-	D-	E-	F-
-0			(SP)	0	@	P	`	p				(RSP)	°	À	Đ	à	ð
-1			!	1	A	Q	a	q			‘	¡	±	Á	Ñ	á	ñ
-2			"	2	B	R	b	r		,	’	¢	²	Â	Ò	â	ò
-3			#	3	C	S	c	s			“	£	³	Ã	Ó	ã	ó
-4	¯		$	4	D	T	d	t		„	”	¤	´	Ä	Ô	ä	ô
-5	˘		%	5	E	U	e	u		...	•	¥	µ	Å	Õ	å	õ
-6	˙		&	6	F	V	f	v		†	–	¦	¶	Æ	Ö	æ	ö
-7			'	7	G	W	g	w		‡	—	§	·	Ç	×	ç	÷
-8	°		(	8	H	X	h	x		^	~	¨	¸	È	Ø	è	ø
-9			)	9	I	Y	i	y		‰	™	©	¹	É	Ù	é	ù
-A	″		*	:	J	Z	j	z		Š	š	ª	º	Ê	Ú	ê	ú
-B	¸		+	;	K	[	k	{		‹	›	«	»	Ë	Û	ë	û
-C	ˇ		,	<	L	\	l	\|		Œ	œ	¬	¼	Ì	Ü	ì	ü
-D			-	=	M	]	m	}				(SHY)	½	Í	Ý	í	ý
-E			.	>	N	^	n	~				®	¾	Î	Þ	î	þ
-F			/	?	O	_	o				Ÿ	¯	¿	Ï	ß	ï	ÿ

Figure 1. DeskTop Publishing: ASCII Code Page 1004

© Copyright IBM Corp. 1990

Font Layout

On page 28-5, replace this section with the following:

The following table lists the full character set in the order in which the characters occur in the multi-code-page font. Characters are listed in order of their universal glyph list (UGL) number; the graphic character global identifier (GCGID) and a description of each character are also given.

UGL	GCGID	Description
1	SS000000	Smiling face
2	SS010000	Smiling face, reverse image
3	SS020000	Heart suit symbol
4	SS030000	Diamond suit symbol
5	SS040000	Club suit symbol
6	SS050000	Spade suit symbol
7	SM570000	Bullet
8	SM570001	Bullet, reverse image
9	SM750000	Open circle
10	SM750002	Open circle, reverse image
11	SM280000	Male symbol
12	SM290000	Female symbol
13	SM930000	Musical note
14	SM910000	Two musical notes
15	SM690000	Sun symbol
16	SM590000	Forward arrow indicator
17	SM630000	Back arrow indicator
18	SM760000	Up-down arrow
19	SP330000	Double exclamation point
20	SM250000	Paragraph symbol (USA)
21	SM240000	Section symbol (USA), paragraph (Europe)
22	SM700000	Solid horizontal rectangle
23	SM770000	Up-down arrow, perpendicular
24	SM320000	Up arrow
25	SM330000	Down arrow
26	SM310000	Right arrow
27	SM300000	Left arrow
28	SA420000	Right angle symbol
29	SM780000	Left-right arrow
30	SM600000	Solid triangle
31	SV040000	Solid triangle, inverted
32	SP010000	Space
33	SP020000	Exclamation point
34	SP040000	Quotation marks, double
35	SM010000	Number sign
36	SC030000	Dollar sign
37	SM020000	Percent sign
38	SM030000	Ampersand
39	SP050000	Apostrophe
40	SP060000	Left parenthesis
41	SP070000	Right parenthesis
42	SM040000	Asterisk
43	SA010000	Plus sign
44	SP080000	Comma
45	SP100000	Hyphen/minus sign
46	SP110000	Period/full stop
47	SP120000	Slash
48	ND100000	Zero

UGL	GCGID	Description
49	ND010000	One
50	ND020000	Two
51	ND030000	Three
52	ND040000	Four
53	ND050000	Five
54	ND060000	Six
55	ND070000	Seven
56	ND080000	Eight
57	ND090000	Nine
58	SP130000	Colon
59	SP140000	Semicolon
60	SA030000	Less than sign/greater than (Arabic)
61	SA040000	Equal Sign
62	SA050000	Greater than sign/less than (Arabic)
63	SP150000	Question mark
64	SM050000	At sign
65	LA020000	A capital
66	LB020000	B capital
67	LC020000	C capital
68	LD020000	D capital
69	LE020000	E capital
70	LF020000	F capital
71	LG020000	G capital
72	LH020000	H capital
73	LI020000	I capital
74	LJ020000	J capital
75	LK020000	K capital
76	LL020000	L capital
77	LM020000	M capital
78	LN020000	N capital
79	LO020000	O capital
80	LP020000	P capital
81	LQ020000	Q capital
82	LR020000	R capital
83	LS020000	S capital
84	LT020000	T capital
85	LU020000	U capital
86	LV020000	V capital
87	LW020000	W capital
88	LX020000	X capital
89	LY020000	Y capital
90	LZ020000	Z capital
91	SM060000	Square left bracket
92	SM070000	Backslash
93	SM080000	Square right bracket
94	SD150000	Circumflex accent
95	SP090000	Underline, continuous underscore
96	SD130000	Grave accent
97	LA010000	a small
98	LB010000	b small
99	LC010000	c small
100	LD010000	d small
101	LE010000	e small
102	LF010000	f small
103	LG010000	g small
104	LH010000	h small
105	LI010000	i small
106	LJ010000	j small
107	LK010000	k small
108	LL010000	l small
109	LM010000	m small

UGL	GCGID	Description
110	LN010000	n small
111	LO010000	o small
112	LP010000	p small
113	LQ010000	q small
114	LR010000	r small
115	LS010000	s small
116	LT010000	t small
117	LU010000	u small
118	LV010000	v small
119	LW010000	w small
120	LX010000	x small
121	LY010000	y small
122	LZ010000	z small
123	SM110000	Left brace
124	SM130000	Vertical line, logical OR
125	SM140000	Right brace
126	SD190000	Tilde
127	SM790000	House
128	LC420000	C cedilla capital
129	LU170000	u diaeresis small
130	LE110000	e acute small
131	LA150000	a circumflex small
132	LA170000	a diaeresis small
133	LA130000	a grave small
134	LA270000	a overcircle small
135	LC410000	c cedilla small
136	LE150000	e circumflex small
137	LE170000	e diaeresis small
138	LE130000	e grave small
139	LI170000	i diaeresis small
140	LI150000	i circumflex small
141	LI130000	i grave small
142	LA180000	A diaeresis capital
143	LA280000	A overcircle capital
144	LE120000	E acute capital
145	LA510000	ae diphthong small
146	LA520000	AE dipthong capital
147	LO150000	o circumflex small
148	LO170000	o diaeresis small
149	LO130000	o grave small
150	LU150000	u circumflex small
151	LU130000	u grave small
152	LY170000	y diaeresis small
153	LO180000	O diaeresis capital
154	LU180000	U diaeresis capital
155	LO610000	o slash small
156	SC020000	Pound sterling sign
157	LO620000	O slash capital
158	SA070000	Multiply sign
159	SC070000	Florin sign
160	LA110000	a acute small
161	LI110000	i acute small
162	LO110000	o acute small
163	LU110000	u acute small
164	LN190000	n tilde small
165	LN200000	N tilde capital
166	SM210000	Ordinal indicator, feminine
167	SM200000	Ordinal indicator, masculine
168	SP160000	Question mark, inverted
169	SM530000	Registered trademark symbol
170	SM660000	Logical NOT, end of line symbol

UGL	GCGID	Description
171	NF010000	One-half
172	NF040000	One-quarter
173	SP030000	Exclamation point, inverted
174	SP170000	Left angled quotes
175	SP180000	Right angled quotes
176	SF140000	Fill character, light
177	SF150000	Fill character, medium
178	SF160000	Fill character, heavy
179	SF110000	Center box bar vertical
180	SF090000	Right middle box side
181	LA120000	A acute capital
182	LA160000	A circumflex capital
183	LA140000	A grave capital
184	SM520000	Copyright symbol
185	SF230000	Right box side double
186	SF240000	Center box bar vertical double
187	SF250000	Upper right box corner double
188	SF260000	Lower right box corner double
189	SC040000	Cent sign
190	SC050000	Yen sign
191	SF030000	Upper right box corner
192	SF020000	Lower left box corner
193	SF070000	Middle box bottom
194	SF060000	Middle box top
195	SF080000	Left middle box side
196	SF100000	Center box bar horizontal
197	SF050000	Box intersection
198	LA190000	a tilde small
199	LA200000	A tilde capital
200	SF380000	Lower left box corner double
201	SF390000	Upper left box corner double
202	SF400000	Middle box bottom double
203	SF410000	Middle box top double
204	SF420000	Left box side double
205	SF430000	Center box bar horizontal double
206	SF440000	Box intersection double
207	SC010000	International currency symbol
208	LD630000	eth Icelandic small
209	LD620000	D stroke capital, Eth Icelandic capital
210	LE160000	E circumflex capital
211	LE180000	E diaeresis capital
212	LE140000	E grave capital
213	LI610000	i dotless small
214	LI120000	I acute capital
215	LI160000	I circumflex capital
216	LI180000	I diaeresis capital
217	SF040000	Lower right box corner
218	SF010000	Upper left box corner
219	SF610000	Solid fill character
220	SF570000	Solid fill character, bottom half
221	SM650000	Vertical line, broken
222	LI140000	I grave capital
223	SF600000	Solid fill character, top half
224	LO120000	O acute capital
225	LS610000	Sharp s small
226	LO160000	O circumflex capital
227	LO140000	O grave capital
228	LO190000	o tilde small
229	LO200000	O tilde capital
230	SM170000	Micro symbol
231	LT630000	Thorn Icelandic small

UGL	GCGID	Description
232	LT640000	Thorn Icelandic capital
233	LU120000	U acute capital
234	LU160000	U circumflex capital
235	LU140000	U grave capital
236	LY110000	y acute small
237	LY120000	Y acute capital
238	SM150000	Overline
239	SD110000	Acute accent
240	SP320000	Syllable hyphen
241	SA020000	Plus or minus sign
242	SM100000	Double underscore
243	NF050000	Three-quarters
244	SM250000	Paragraph symbol (USA)
245	SM240000	Section symbol (USA), paragraph (Europe)
246	SA060000	Divide sign
247	SD410000	Cedilla (or sedila) accent
248	SM190000	Degree symbol
249	SD170000	Diaeresis, umlaut accent
250	SD630000	Middle dot
251	ND011000	One superscript
252	ND031000	Three superscript
253	ND021000	Two superscript
254	SM470000	Solid square, histogram, square bullet
255	SP300000	Required space
256	SC060000	Peseta sign
257	SM680000	Start of line symbol
258	SF190000	Right box side double to single
259	SF200000	Right box side single to double
260	SF210000	Upper right box corner single to double
261	SF220000	Upper right box corner double to single
262	SF270000	Lower right box corner single to double
263	SF280000	Lower right box corner double to single
264	SF360000	Left box side single to double
265	SF370000	Left box side double to single
266	SF450000	Middle box bottom single to double
267	SF460000	Middle box bottom double to single
268	SF470000	Middle box top double to single
269	SF480000	Middle box top single to double
270	SF490000	Lower left box corner double to single
271	SF500000	Lower left box corner single to double
272	SF510000	Upper left box corner single to double
273	SF520000	Upper left box corner double to single
274	SF530000	Box intersection single to double
275	SF540000	Box intersection double to single
276	SF580000	Solid fill character, left half
277	SF590000	Solid fill character, right half
278	GA010000	Alpha small
279	GG020000	Gamma capital
280	GP010000	Pi small
281	GS020000	Sigma capital
282	GS010000	Sigma small
283	GT010000	Tau small
284	GF020000	Phi capital
285	GT620000	Theta capital
286	GO320000	Omega capital
287	GD010000	Delta small
288	SA450000	Infinity symbol
289	GF010000	Phi small
290	GE010000	Epsilon small
291	SA380000	Intersection, logical product
292	SA480000	Indentity symbol, almost equal

UGL	GCGID	Description
293	SA530000	Greater than or equal sign
294	SA520000	Less than or equal sign
295	SS260000	Upper integral symbol section
296	SS270000	Lower integral symbol section
297	SA700000	Nearly equals symbol
298	SA790000	Product dot
299	SA800000	Radical symbol
300	LN011000	n small superscript
301	SD310000	Macron accent
302	SD230000	Breve accent
303	SD290000	Overdot accent (over small Alpha)
304	SD270000	Overcircle accent
305	SD250000	Double acute accent
306	SD430000	Ogonek accent
307	SD210000	Caron accent
308	SP190000	Left single quote
309	SP200000	Right single quote
310	SP210000	Left double quotes
311	SP220000	Right double quotes
312	SS680000	Endash
313	SM900000	Emdash
314	SD150000	Circumflex accent
315	SD190000	Tilde accent
316	SP260000	Single quote on baseline (German lower)
317	SP230000	Left lower double quotes
318	SV520000	Ellipsis
319	SM340000	Dagger footnote indicator
320	SM350000	Double dagger footnote indicator
321	SD150100	Circumflex accent (over small alpha)
322	SM560000	Permille symbol
323	LS220000	S caron capital
324	SP270000	French single open quote
325	LO520000	OE ligature capital
326	SD190100	Tilde accent (over small alpha)
327	SM540000	Trademark symbol
328	LS210000	s caron small
329	SP280000	French single close quote
330	LO510000	oe ligature small
331	LY180000	Y diaeresis capital

Appendix B. Standard Bit-Map Formats

Summary of Changes

New	Updated	Section Title
	√	Bit-Map File Format

Bit-Map File Format

On pages B-2 and B-3, replace the text with the following:

Presentation Manager uses the same file format for bit maps, icons, and pointers in resource files. In the following description, "bit map" refers to bit maps, icons, and pointers unless otherwise specified.

Two formats are supported. In the first, a single-size version of the bit map is defined. This is used whatever the target device.

The second format allows multiple versions of the bit map to be defined, including one or more device-independent versions, and a number of device-dependent versions each intended for use with a particular device.

In the case of icons and pointers, when more than one version of the bit map exists, the preferred version is one that matches the device size of icon or pointer. Otherwise, the device-independent version is used to scale a bit map to the required size.

For general bit maps, which may be of arbitrary size, the preferred version is the one matching the requested bit map size; otherwise, the one matching the display size is selected. If neither is available, the device-independent version is used from which to scale a bit map.

For both formats, the definition consists of two sections. The first section contains general information about the type, dimensions, and other attributes of the resource. The second section contains data describing the pels that make up the bit map(s), and is in the format specified in "Bit-Map Data" on page B-1.

In the multiple-version format, the first section contains an array of BITMAPARRAYFILEHEADER structures. The format of this is as follows:-

```
typedef struct _BITMAPARRAYFILEHEADER { /* bafh */
    USHORT            usType;
    ULONG             cbSize;
    ULONG             offNext;
    USHORT            cxDisplay;
    USHORT            cyDisplay;
    BITMAPFILEHEADER  bfh;
} BITMAPARRAYFILEHEADER;
typedef BITMAPARRAYFILEHEADER FAR *PBITMAPARRAYFILEHEADER;
```

The fields in BITMAPARRAYFILEHEADER have these meanings:

usType Type of structure. This is:

© Copyright IBM Corp. 1990

BFT_BITMAPARRAY (X'4142' - 'BA' for BITMAPARRAYFILEHEADER)

cbSize Size of the BITMAPARRAYFILEHEADER structure in bytes.

offNext Offset of the next BITMAPARRAYFILEHEADER structure from the start of the file

cxDisplay, cyDisplay Pel dimensions of the device for which this version is intended (for example, 640 x 480 for VGA).

The device-independent version must be the first BITMAPARRAYFILEHEADER defined.

In the single-size format, the BITMAPARRAYFILEHEADER structure is not present. The definition consists of one or two BITMAPFILEHEADER structures.

The format of the BITMAPFILEHEADER structure is :

```
typedef struct _BITMAPFILEHEADER { /* bfh */
    USHORT            usType;
    ULONG             cbSize;
    INT               xHotspot;
    INT               yHotspot;
    ULONG             offBits;
    BITMAPINFOHEADER  bmp;
} BITMAPFILEHEADER;
typedef BITMAPFILEHEADER FAR *PBITMAPFILEHEADER;
```

BITMAPINFOHEADER is a standard data type (see above, and also *BITMAPINFOHEADER* on page 2-3.

The fields in BITMAPFILEHEADER have these meanings:

usType Type of resource the file contains. The valid values are:

BFT_BMAP (X'4D42' - 'BM' for bit maps)
BFT_ICON (X'4349' - 'IC' for icons)
BFT_POINTER (X'5450' - 'PT' for pointers).
BFT_COLORICON (X'4943' - 'CI' for color icons).
BFT_COLORPOINTER (X'5043' - 'CP' for color pointers).

cbSize Size of the BITMAPFILEHEADER structure in bytes.

xHotspot, yHotspot Coordinates of the hotspot for icons and pointers. This field is ignored for bit maps.

offBits Offset in bytes to the beginning of the bit-map pel data in the file, from the start of the definition.

For icons and pointers, the **bitmapheight** field in **bmp** is actually twice the pel height of the image that appears on the screen. This is because these types actually contain two full bit-map pel definitions. The first bit-map definition is the XOR mask, which contains invert information (0 = no invert, 1 = invert) for the pointer or icon. The second is the AND mask, which determines whether the pointer or the screen is shown (0 = black/white, 1 = screen/inverse screen).

For color icons or pointers, there are two bit-maps involved: one that is black and white and consists of an AND and an XOR mask, and one that is color that defines the color content.

The **bitmapheight** field in the *BITMAPINFOHEADER* structure for the color bit-map must be the real height, that is, half the value specified for the black and white bit-map. The **bitmapwidth** fields must be the same.

The following table shows how these two bit-maps are used for a color icon or pointer:

XOR	AND	COLOR	
1	1	x	Invert screen
0	0	x	Use color x
0	1	x	Transparency
1	0	x	Use color x

For color icons or pointers, two BITMAPFILEHEADER structures are therefore required:

```
BITMAPFILEHEADER     with usType BFT_COLORICON or BFT_COLORPOINTER
   BITMAPINFOHEADER  (part of BITMAPFILEHEADER)
   Color table
BITMAPFILEHEADER     with same usType
   BITMPAINFOHEADER  (part of BITMAPFILEHEADER)
   Color table
..
bits for one bit-map
..
..
bits for other bit-map
..
```

The usType for the first BITMAPFILEHEADER is either BFT_COLORICON or BFT_COLORPOINTER. This means that a second BITMAPFILEHEADER is present as part of the definition of a color icon or pointer. The first BITMAPFILEHEADER structure contains the information for the black and white AND and XOR masks, while the second BITMAPFILEHEADER structure contains the information for the color part of the pointer or icon.

For the multiple version format, the file is as follows:

```
BITMAPARRAYFILEHEADER    for device-independent version
   BITMAPFILEHEADER      (part of BITMAPARRAYFILEHEADER)
      BITMPAINFOHEADER   (part of BITMAPFILEHEADER)
      Color table

   BITMAPFILEHEADER      )
      BITMAPINFOHEADER   )  only if this is a color icon or pointer
      Color table        )

BITMAPARRAYFILEHEADER    for first device-dependent version
   BITMAPFILEHEADER      (part of BITMAPARRAYFILEHEADER)
      BITMPAINFOHEADER   (part of BITMAPFILEHEADER)
      Color table

   BITMAPFILEHEADER      )
      BITMAPINFOHEADER   )  only if this is a color icon or pointer
      Color table        )

Further BITMAPARRAYFILEHEADER groups occur here as required
for additional device-dependent versions

..
bits for one bit-map
..
..
bits for next bit-map
..

And so on for as many bit-maps as necessary.
```

Appendix D. Format of Interchange Files

Summary of Changes

New	Updated	Section Title
	√	Metafile Restrictions

Metafile Restrictions

On page D-1, replace this section with the following. Changes are indicated by a vertical bar.

The following restrictions apply to the generation of all metafiles, and also to the generation of a PM_Q_STD print file to a OD_QUEUED device:

- If a bit map is used as the source of a GpiWCBitBlt or GpiBitBlt operation, or as an area-fill pattern, it must not be modified or deleted (GpiDeleteBitmap) before the metafile is closed.
- GpiSetPS must not be used.
- GpiSetPageViewport is ignored.

The following section lists some general rules that must be followed when creating a metafile that is to be acceptable to SAA-conforming implementations, or replayed into a presentation space that is in draw-and-retain or retain mode (see GpiSetDrawingMode).

- These items must be established or defaulted before any drawing occurs to the graphics presentation space, and not changed subsequently:
 - The graphics field (GpiSetGraphicsField). For an SAA-conforming metafile, the graphics field must be defaulted or set to no clipping.
 - The code page for the default character set (GpiSetCp).
 - The color table (GpiCreateLogColorTable). The size of the color table must not exceed 31KB (KB equals 1024 bytes).
 - The default viewing transform (GpiSetDefaultViewMatrix).
 - The setting of the draw controls (GpiSetDrawControl). DCTL_DISPLAY must be defaulted or set ON.
 - The default values of attributes (see GpiSetDefAttrs), viewing limits (see GpiSetDefViewingLimits), primitive tag (see GpiSetDefTag) and arc parameters (see GpiSetDefArcParams).

 These items must not be changed while the metafile context remains open.
- These calls must not be used:
 - GpiBitBlt
 - GpiDeleteSetId (note that this means that local identifiers cannot be used again within the picture)
 - GpiErase
 - GpiExcludeClipRectangle
 - GpiIntersectClipRectangle
 - GpiOffsetClipRegion
 - GpiPaintRegion
 - GpiResetPS

© Copyright IBM Corp. 1990

 - GpiSetClipRegion
 - GpiSetPel
 - GpiSetPS
 - DevEscape (for an escape which is metafiled).
- The metafile context must not be reassociated.
- If a bit map is used as the source of a GpiWCBitBlt operation, or as an area-fill pattern, it must not be modified or deleted (GpiDeleteBitmap) before the metafile is closed.
- Only these foreground mixes must be used (see GpiSetMix):
 - FM_DEFAULT
 - FM_OR
 - FM_OVERPAINT
 - FM_LEAVEALONE.
- Only these background mixes must be used (see GpiSetBackMix):
 - BM_DEFAULT
 - BM_OVERPAINT
 - BM_LEAVEALONE.

Programming Note: There is no restriction concerning the use of primitives outside segments. These are metafiled in segment(s) with zero identifier.

Master Index

© Copyright IBM Corp. 1990

Index

A

Access a DRAGINFO Structure 48
Access Drag Information 34
Add Print Destination 80
Add Print Queue 97
Allocate DRAGINFO Structure 36
Allocate DRAGTRANSFER Structures 37

B

Begin Dragging Files 42
Bit-Map File Format 189
BITMAPINFO 3
BITMAPINFOHEADER 4
Button Control Styles 133

C

CFI_* values 121
clipping 193
Code Pages 181
color table 193
Combo Box Control Window Messages 145
Continue Print Job 89
Continue Print Queue 99
Control Print Destination 82
Control Statements, Predefined 179
Create String Handle 35

D

DC_* values 7
Delete DRAGINFO String Handles 38
Delete Print Destination 83
Delete Print Job 90
Delete Print Queue 100
Delete String Handle 39
DevEscape 29
DevOpenDC 29
Direct Manipulation for Files 32
Directives 179
DM_DRAGERROR 161
DM_DRAGFILECOMPLETE 162
DM_DRAGLEAVE 163
DM_DRAGOVER 163
DM_DROP 165
DM_DROPHELP 166
DM_EMPHASIZETARGET 166
DM_ENDCONVERSATION 167
DM_FILERENDERED 168
DM_PRINT 169
DM_RENDER 169
DM_RENDERCOMPLETE 170
DM_RENDERFILE 171
DM_RENDERPREPARE 172
DosPrintDestAdd 80
DosPrintDestControl 82
DosPrintDestDel 83
DosPrintDestEnum 84
DosPrintDestGetInfo 85
DosPrintDestSetInfo 87
DosPrintJobContinue 89
DosPrintJobDel 90
DosPrintJobEnum 91
DosPrintJobGetId 92
DosPrintJobGetInfo 93
DosPrintJobPause 94
DosPrintJobSetInfo 95
DosPrintQAdd 97
DosPrintQContinue 99
DosPrintQDel 100
DosPrintQEnum 101
DosPrintQGetInfo 103
DosPrintQPause 105
DosPrintQPurge 106
DosPrintQSetInfo 107
Down cursor key 123
DO_* values 5, 7
Drag 40
drag information
 access 34
DRAGIMAGE 4
DRAGINFO 5
DRAGITEM 5
DRAGTRANSFER 7
DRF_* values 6
DrgAcceptDroppedFiles 32
DrgAccessDraginfo 34
DrgAddStrHandle 35
DrgAllocDraginfo 36
DrgAllocDragtransfer 37
DrgDeleteDraginfoStrHandles 38
DrgDeleteStrHandle 39
DrgDrag 40
DrgDragFiles 42
DrgFreeDraginfo 44
DrgFreeDragtransfer 45
DrgGetPS 46
DrgPostTransferMsg 47
DrgPushDraginfo 48
DrgQueryDragitem 49
DrgQueryDragitemCount 50
DrgQueryDragitemPtr 51
DrgQueryNativeRMF 52
DrgQueryNativeRMFLen 53
DrgQueryStrName 54

© Copyright IBM Corp. 1990

DrgQueryStrNameLen 55
DrgQueryTrueType 56
DrgQueryTrueTypeLen 57
DrgReleasePS 58
DrgSendTransferMsg 59
DrgSetDragImage 60
DrgSetDragItem 61
DrgSetDragPointer 62
DrgVerifyNativeRMF 63
DrgVerifyRMF 64
DrgVerifyTrueType 65
DrgVerifyType 66
DrgVerifyTypeSet 67
DRG_* values 4
DRM_* values 6
DRT_* values 5

E

Enter key 123
Entry Field Control Styles 135
Enumerate Print Destination 84
Enumerate Print Job 91
Enumerate Print Queue 101
Esc key 124
ESCSETMODE 8

F

Font Layout 182
FONTMETRICS 9
Free DRAGINFO structure 44
Free DRAGTRANSFER Storage 45
Full Screen VIO Applications 181

G

Get Drag Presentation Space 46
Get Dragged Object Count 50
Get DRAGITEM Structure 49, 51
Get Format of a Dragged Object 52
Get Print Destination Information 85
Get Print Job ID 92
Get Print Job Information 93
Get Print Queue Information 103
Get String Contents 54
Get String Length 55
Get String Length of a Dragged Object 53
Get String Length of Dragged Object 57
Get True Type of Dragged Object 56
GpiAssociate 69
GpiCharStringPos 69
GpiCharStringPosAt 70
GpiCreateLogColorTable 70
GpiCreateLogFont 70
GpiDeleteBitmap 70
GpiPlayMetafile 71
GpiQueryBitmapParameters 71
GpiQueryCharStringPos 71
GpiQueryCharStringPosAt 71
GpiQueryDefCharBox 71
GpiQueryLogColorTable 71
GpiQueryTextBox 72
GpiRealizeColorTable 72
GpiResetPS 72
GpiSetCharBox 72
GpiSetClipRegion 73
GpiSetMarkerSet 73
GpiSetMetaFileBits 73
GpiSetPageViewport 73
GpiSetPatternSet 73
GpiUnrealizeColorTable 73

H

HFILE 15
HM_ACTION_BAR_COMMAND 177
HSTR 15

J

JournalPlaybackHook 129

K

kerning 15
 number of pairs 15

L

Left cursor key 123
List Box Control Styles 139
LM_QUERYCURSOR 139
LM_SETCURSOR 139
LM_SETSELECTION 140
LS_NOVERTSCROLL 139

M

MARGSTRUCT 16
Menu Template 179
Metafile Restrictions 193
MLE_SEARCHDATA 16
MLM_FORMAT 143
MLM_QUERYSEL 143
MLM_SEARCH 144
MLM_SETFORMATRECT 144
MM_DISMISSMENU 141
MM_QUERYITEMTEXT 141
MM_SELECTITEM 141
MsgFilterHook 129
MT 16

O

OVERFLOW 17

P

PAPSZ 17
Pause Print Job 94
Pause Print Queue 105
PicIchg 75
PicPrint 75
Post Drag Message 47
PRDINFO 17
PRDINFO3 18
Predefined Control Statements 179
PrfQueryProgramTitles 77
PrfReset 77
PRIDINFO 19
PRJINFO 19
PRJINFO2 20
PRJINFO3 22
PRQINFO 24
PRQINFO3 26
Purge Print Queue 106

R

Release Presentation Space 58
RENDERFILE 27
Resource Files 179
Right cursor key 123

S

SBM_QUERYHILITE 147
SBM_SETHILITE 147
SBM_SETSCROLLBAR 147
Send Drag Message 59
Set Drag Image 60
Set Mouse Pointer 62
Set Print Destination Information 87
Set Print Job Information 95
Set Print Queue Information 107
Set Values in DRAGITEM 61
SPBM_OVERRIDESETLIMITS 150
SPBM_QUERYLIMITS 151
SPBM_QUERYVALUE 152
SPBM_SETARRAY 153
SPBM_SETCURRENTVALUE 154
SPBM_SETLIMITS 155
SPBM_SETMASTER 156
SPBM_SETTEXTLIMIT 157
SPBM_SPINDOWN 158
SPBM_SPINUP 158
spin button control styles 149
spin button control window processing 149
SPLERR 27
spooler
 continue queue 99
spooler *(continued)*
 control device 82
 create device 80
 create queue 97
 delete device 83
 delete job 90
 delete queue 100
 enumerate device 84
 enumerate job 91
 enumerate queue 101
 get job ID 92
 hold job 94
 hold queue 105
 purge queue 106
 query device 85
 query job 93
 query queue 103
 release job 89
 set device 87
 set job information 95
 set queue 107
string handle
 create 35
 delete 38, 39

U

Up cursor key 123

V

Verify Rendering Mechanism and Format 63, 64
Verify True Type of Dragged Object 65
Verify Type of Dragged Object 66
Verify Types 67
VioQueryFonts 109
VioSetDeviceCellSize 109

W

WinAddSwitchEntry 114
WinAllocMem 114
WinBeginPaint 114
WinCalcFrameRect 114
WinChangeSwitchEntry 114
WinCopyRect 115
WinCreateCursor 115
WinCreateHeap 115
WinCreateMsgQueue 115
WinCreateStdWindow 115
WinCreateSwitchEntry 115
WinDestroyWindow 116
window processing
 spin button control 149
WinDrawBitmap 116
WinDrawBorder 116
WinDrawText 116

WinEnumDlgItem 116
WinEqualRect 116
WinFillRect 117
WinFreeMem 117
WinGetErrorInfo 117
WinInflateRect 117
WinInitialize 117
WinIntersectRect 117
WinInvalidateRect 118
WinInvertRect 118
WinIsRectEmpty 118
WinLoadDlg 118
WinLockVisRegions 118
WinMessageBox 118
WinOffsetRect 119
WinPtInRect 119
WinQueryQueueStatus 119
WinQuerySysValue 119
WinQueryUpdateRect 119
WinQueryWindowRect 120
WinQueryWindowUshort 120
WinReallocMem 120
WinRegisterClass 120
WinScrollWindow 120
WinSetClassMsgInterest 120
WinSetClipbrdData 121
WinSetHook 121
WinSetMsgInterest 122
WinSetRect 122
WinSetRectEmpty 122
WinSetSysValue 122
WinSubtractRect 122
WinTrackRect 123
WinUnionRect 125
WinValidateRect 125
WM_CONTROL 143, 145
WM_CONTROL (in Spin Controls) 159
WM_DDE_INITIATE 175
WM_DDE_INITIATEACK 175
WM_DRAWITEM 139
WM_FORMATFRAME 137
WM_HITTEST 131
WM_MEASUREITEM 139
WM_SETSELECTION 131
WM_UPDATESTYLE 145

IBM United Kingdom
International Products Limited
PO Box 41, North Harbour
Portsmouth, PO6 3AU
England

Printed in Denmark
Scanprint a-s, Viby J

